Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2011

S. Farrell
Trinity College Dublin
C. Dannewitz
University of Paderborn
B. Ohlman
Ericsson
D. Kutscher
NEC
March 28, 2011

URIs for Named Information
draft-farrell-ni-00

Abstract

This document defines a URI-based name form for objects intended to be used for information-centric networking and more generally. The name form defined here allows for the various forms of hash-based binding between the name and the named-object, as well as supporting human-readable and hierarchical names.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 29, 2011.

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

[[Text in double square brackets (like this) is commentary.]]

URIs [RFC3986] are used in various protocols for identifying resources. In many deployments those URIs contain strings that are hash function outputs in order to ensure uniqueness in terms of mapping the URI to a specific resource, or to make URIs hard to guess for security reasons. However, there is no standard way to interpret those strings and so today in general only the creator of the URI knows how to use the hash function output.

In the context of information-centric networking [ref.netinf-design] [ref.ccn] and elsewhere there is value in being able to compare a presented resource against the URI that was de-referenced in order to access that resource. If a cryptographically-strong comparison function can be used then this allows for many forms of in-network storage, without requiring as much trust in the infrastructure used to present the resource. The outputs of hash functions can be used in this manner, if presented in a standard way. There are also many other potential uses for these hash outputs, for example, in terms of binding the URI to an owner via signatures and public keys, mapping between names, handling versioning etc. Many such uses can be based on "wrapping" the object with meta-data, e.g. including signatures, public key certificates etc.

We therefore define the "ni" URI scheme that allows for, but does not insist upon, checking of the integrity of the URI/resource mapping.

Hash-function outputs however are not human memorable, and cannot easily be used to construct a hierarchical namespace, which some protocols and applications may require. The URI scheme therefore also allows for human-readable strings to be used with, or instead of, the hash function output strings.

We expect it will be beneficial for applications to be able to map

between human-readable URIs and URIs that allow for validation of integrity the URI/resource mapping. However, in order to keep our scheme simple and more broadly applicable, all considerations for how to map between URIs and for how to access resources using these URIs are to be specified elsewhere. In this memo, we simply define a form of URI that can be used hopefully in many different contexts.

The URI scheme defined here could be thought of as being similar to URLs with the ability to verify the URL/resource mapping. However, we envisage these URIs actually being of most use in applications where the resource is not located at a particular place in a network topology, but can rather be cached in many places.

Syntax definitions in this memo are specified according to ABNF [RFC4648].

2. Hash Strings

We start with specifying how the outputs from hash functions are handled.

Hash outputs are binary values that MUST be base64 encoded with line-feeds, spaces and terminating "=" characters removed. These values MUST be immediately preceded with a hash algorithm identifier and a separator character (":"). For example, the start of such a value might look like: "sha256:NDc0NzgyMGVmOGQ3OGU0..."

Hash values MAY be followed by a function identifier, naming the function to be used to verify that hash. If the function identifier is omitted, then the application needs to know how to verify the URI/resource mapping if that is desired.

In many cases the input to the hash function will the actual resource itself as presented by whatever protocol uses the name and this is the default when the function identifier is omitted. This is what would be in the body of a HTTP response, were the URI used in a HTTP GET message and were the object returned in a 200 OK HTTP response with no fragmentation.

The function identifier allows for cases where the resource is actually presented with additional information (e.g. meta-data) or is

wrapped in other encoding. One way in which this is expected to be used is when the resource is presented with an accompanying digital signature. In that case the signature could be presented along with the resource and the hash function could be calculated over some combination of the resource and signature, or, just over the signature bits. (Note that the signature bits themselves are not part of the name in this example.)

Since we want to be able to verify the hash value against the resource, and since sometimes this will involve the resource being wrapped in some other format that allows inclusion of meta-data or security data, it may be the case that the protocol that presents the resource identifies it as having the "wrapped" type. In order to support applications that require typing for the resource itself (as opposed to its "wrapped" form) we also allow a hash value to be accompanied with an "inner" type that identifies the type of the resource, rather than the wrapper. We do this using MIME types appended after the function identifier.

Note that the "/" character from the MIME type MUST be percent encoded in order to conform to the ABNF below. That is "application/jpeg" will be presented as "application%2fjpeg".

[[Should or must we allow "%2F" as well?]]

The "default" function-identifier, which is the only one defined here, is denoted with the string "id" and means that the resource when returned can be directly fed into the hash function without any canonicalization required, so this is the "identity" function. Of course, the hash based comparison may fail if some middlebox or access protocol has re-encoded the resource in some way.

The "id" function identifier can be used if an "inner" MIME type should be added to the name.

[["id" may not be the best tag for this, since it may confuse, not sure what else to use.]]

Hash algorithm identifiers and function identifiers are to be registered, in an IANA registry (see the "IANA Considerations" section below).

[[There may already be a usable hash function registry. But if we're going to be interested in truncated hashes then we may need our own.]]

Let's call a value encoded as above a "hash-string." We could define it thusly:

```
hash-string = hashalg ":" b64value
                  [ ":" function-identifier [ ":" mime-type ] ]

hashalg = identifier

function-identifier = identifier

identifier = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

mime-type = type %2f subtype
```

b64value is a string based on the 64-character subset of US-ASCII as defined in [RFC4648]. mime-type is based on the Content-Type header field syntax as specified in [RFC2045], but using %2f as a delimiter between type and subtype instead of "/", and without parameters.

[[Complete formal ABNF spec of b64value to be provided in a future revision of this memo.]]

It is important to note that implementations are NOT REQUIRED to support any cryptographic operations, that is, as necessary, they need to be able to parse, route, log, and resolve names with any of the above fields, but do not have to verify anything cryptographic.

Implementations that do support cryptographic operations MUST offer applications a way (e.g. via an API) to compare an ni name with a resource. The set of cryptographic operations to be supported (e.g. the set of supported function identifiers), is an implementation decision and is not further specified. Where an implementation does not support the operation needed to verify a ni object, it MUST return an error distinct from the case where the name-to-object comparison failed, e.g. due to a hash mismatch.

Implementations that create names such as these MUST ensure that it

is possible to validate the mapping from the name to the resource, should other implementations choose to do that validation. That is, when creating a name like this, make sure that you do it right!

Note that not all protocols and applications making use of this URI form will require strong integrity assurances when doing name/resource comparisons. For this reason, we expect it to be relatively common to use truncated hashes in URIs.

3. URI Scheme

Our URIs consist of the scheme, an optional authority part and then a "local" part which is a possibly empty sequence of either hash-strings or any other string who's encoding is allowed. As with the local part, the authority part may be either a hash-string or any other string who's encoding is allowed.

The semantics of the authority part are not further defined here, but MUST be specified by any protocol or application making use of these URIs.

The "local" part is intended to contain an identifier for the resource in question that is meaningful in the context of the authority.

Note that where the authority part is omitted and where the local part is not a hash-string, then this may be a significant probability for accidental name collisions. Protocols and applications using this URI scheme MUST take care of such collisions, if they matter. Note that this is also true, even if the authority part is present, unless there is some strict authority-part registration scheme in force and where spoofing is hard.

One obvious thing to use for the authority part is a Fully Qualified Domain Name (FQDN), possibly with a port number, in which case applications using these URIs could make use of the Domain Name System (DNS) and TCP. Again though - such uses are outside the scope of this specification. In general, there will be no guarantee that the resource can be accessed at that host:port even in that case.

[[ABNF to be fixed later. Note that the syntax below allows "ni:///"

as a valid name - whether that is good or bad, and if good, what "ni:///" might mean is for future study.]]

```
ni-name = scheme ":" hier-part
hier-part = "//" [authority] "/" *(local-part "/") ["/"]

scheme = "ni"

authority = hash-string | other-string ;(delimiters %-encoded)

local-part = hash-string | other-string ;(delimiters %-encoded)
```

[[Formal ABNF of other-string to be specified in a future revision of this memo.]]

4. Examples

The longer examples in this section flow over lines, but the meaning should be clear enough.

1) ni://tcd.ie/cs8053-exam-2012

Example 1 is quite like a HTTP URL, and simply shows that "normal" URI forms can be used with ni names.

2) ni:///weather-in-dublin-today

Example 2 shows an example of an "intentional" name, where the resource returned will likely change from time to time. This example has no authority part, which presumably would mean that the requester doesn't really care much about the source of the weather information.

3) ni://tcd.ie/sha256:NDVmZTMzOGVkY2JjZGQ0ZmNmZGFlODQ5MjkyZDM0ZTg2ZDI5YzllMmU5OTFlNmE2Mjc3ZTFhN2JhNmE4ZjVmMwo

4) ni:///sha256:NDVmZTMzOGVkY2JjZGQ0ZmNmZGFlODQ5MjkyZDM0ZTg2ZDI5YzllMmU5OTFlNmE2Mjc3ZTFhN2JhNmE4ZjVmMwo

Examples 3 and 4 are the same, one with, and one without, an

authority part. In both cases, we have no idea what was hashed if we only know the ni name. Some higher layer protocol may of course be able to understand what's going on.

It may be that the authority part of example 3 allows for more scalable name-based routing of a request to get, or do something with, that resource.

5) ni://sha256:NDc0NzgyMGVmOGQ3OGU0MmI2MWYwZjY3MDAzNDJmZTY
0NzhhMGY0OTBhMDRiNzA0YTY0MWY0MzVkODQzZWUxMAo:id:sshpk/thing

The authority for example 5 is a hash-string of a public key as stored in a file by openssh. Though we know that from the URI, there is no implication that we need to, or can, do anything special about that fact. Some protocol making use of this name however, might expect that the resource contain a signature verifiable with a public key that matches that hash.

6) ni://tcd.ie/sha256:NDVmZTMzOGVkY2JjZGQ0ZmNmZGFlODQ
5MjkyZDM0ZTg2ZDI5YzllMmU5OTFlNmE2Mjc3ZTFhN2JhNmE4Z
jVmMwo:signeddata:application%2Fjpeg

Example 6 is a ni name for a jpeg file that contains a hash of the file contents where we expect to receive the image in a CMS SignedData wrapper.

Note that the function identifier signeddata could be defined to also accept a PGP or XMLDSIG or other wrapper - what's identified is a function, and not directly a format. The signeddata function is something that would have to be defined elsewhere. That is, another specification would need to be written for each such function. [[Or, something basic might be included here, but not as a mandatory-to-implement feature.]]

5. Security Considerations

[[More needed for sure.]]

Network elements that do attempt to verify the mapping from the name to the resource are doing more work that those that don't, both in terms of CPU, (for the hash and function identifier calculations) and possibly also in terms of network access and/or storage, since they need the resource, and possibly meta-data that might have to be separately requested. An example of the latter kind of meta-data might be a public key certificate or CRL. This additional load could be leveraged in some kinds of DoS attack. Protocols that call for

validation of the name/resource mapping SHOULD specify how to handle any such DoS that may be relevant.

6. IANA Considerations

Two registries will be required for this specification. The first will be for function identifiers, with a FCFS update rule and one initially registered value, the "id" function identifier. The second registry will be for hash functions and may exist already. If not, then, we will want a hash function registry with RFC required as the update rule. Most likely the only reason a new hash function registry would be required would be if we wanted a few relatively weak truncated hash functions registered, but where that would be wrong for the existing hash function registries.

7. Acknowledgements

This work has been supported by the EU FP7 project SAIL. The authors would like to thank SAIL participants to our naming discussions, especially Jean-Francois Peltier, for their input.

8. References

8.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

8.2. Informative References

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[ref.caw] Chapweske, "HTTP Extensions for a Content-Addressable Web", October 2001.

http://lists.w3.org/Archives/Public/www-talk/2001NovDec/0090.html

[ref.ccn] Jacobsen, K, D, F, H, and L, "Networking Named Content", CoNEXT 2009 , December 2009.

[ref.magnet] Mohr, "MAGNET", June 2002.

http://magnet-uri.sourceforge.net/magnet-draft-overview.txt

[ref.netinf-design] Ahlgren, D'Ambrosio, Dannewitz, Marchisio, Marsh, Ohlman, Pentikousis, Rembarz, Strandberg, and Vercellone, "Design Considerations for a Network of Information", Re-Arch 2008 Workshop , December 2008.

Authors' Addresses

Stephen Farrell
Trinity College Dublin
Dublin, 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie

Christian Dannewitz

University of Paderborn
Paderborn
Germany

Email: cdannewitz@upb.de

Borje Ohlman
Ericsson
Stockholm S-16480
Sweden

Email: Borje.Ohlman@ericsson.com

Dirk Kutscher
NEC
Kurfuersten-Anlage 36
Heidelberg,
Germany

Phone:
Email: kutscher@neclab.eu