

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: February 4, 2013

S. Farrell  
Trinity College Dublin  
D. Kutscher  
NEC  
C. Dannewitz  
University of Paderborn  
B. Ohlman  
A. Keranen  
Ericsson  
P. Hallam-Baker  
Comodo Group Inc.  
August 3, 2012

Naming Things with Hashes  
draft-farrell-decade-ni-10

Abstract

This document defines a set of ways to identify a thing (a digital object in this case) using the output from a hash function, specifying a new URI scheme for this, a way to map those to http URLs, and binary and human "speakable" formats for these names. The various formats are designed to support, but not require, a strong link to the referenced object such that the referenced object may be authenticated to the same degree as the reference to it. This work is motivated as a way to standardise current uses of hash outputs in URLs and to support new information-centric applications and other uses of hash outputs in protocols.

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 February 4, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Table of Contents

1. Introduction . . . . .	3
2. Hashes are what Count . . . . .	4
3. Named Information (ni) URI Format . . . . .	6
3.1. Content Type Query String Attribute . . . . .	8
4. .well-known URI . . . . .	9
5. URL Segment Format . . . . .	10
6. Binary Format . . . . .	10
7. Human-speakable (nih) URI Format . . . . .	11
8. Examples . . . . .	12
8.1. Hello World! . . . . .	13
8.2. Public Key Examples . . . . .	13
8.3. nih Usage Example . . . . .	14
9. IANA Considerations . . . . .	15
9.1. Assignment of ni URI Scheme . . . . .	15
9.2. Assignment of nih URI Scheme . . . . .	15
9.3. Assignment of .well-known 'ni' URI . . . . .	16
9.4. Creation of Named Information Hash Algorithm Registry . . . . .	16
9.5. Creation of Named Information Parameter Registry . . . . .	17
10. Security Considerations . . . . .	18
11. Acknowledgments . . . . .	20
12. References . . . . .	20
12.1. Normative References . . . . .	20
12.2. Informative References . . . . .	21
Authors' Addresses . . . . .	22

## 1. Introduction

Identifiers -- names or locators -- are used in various protocols to identify resources. In many scenarios, those identifiers contain values that are obtained from hash functions. Different deployments have chosen different ways to include the hash function outputs in their identifiers, resulting in interoperability problems.

This document defines a "Named Information" identifier, which provides a set of standard ways to use hash function outputs in names. We begin with a few example uses for various ways to include hash function output in a name, with the specifics defined later in this document. Figure 1 shows an example of the Named Information (ni) URI scheme that this document defines.

```
ni:///sha-256;UyaQV-Ev4rdLoHyJJWCillOHfrYv9ElaGQAlMO2X_-Q
```

Figure 1: Example ni URI

Hash function outputs can be used to ensure uniqueness in terms of mapping URIs [RFC3986] to a specific resource, or to make URIs hard to guess for security reasons. Since there is no standard way to interpret those strings today, in general only the creator of the URI knows how to use the hash function output. Other protocols, such as application layer protocols for accessing "smart objects" in constrained environments also require more compact (e.g., binary) forms of such identifiers. In yet other situations people may have to speak such values, e.g., in a voice call, (see Section 8.3), in order to confirm the presence or absence of a resource.

As another example, protocols for accessing in-network storage servers need a way to identify stored resources uniquely and in a location-independent way so that replicas on different servers can be accessed by the same name. Also, such applications may require verification that a resource representation that has been obtained actually corresponds to the name that was used to request the resource, i.e., verifying the binding between the data and the name, which is here termed name-data integrity.

Similarly, 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 used 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 they are presented in a standard way.

Additional applications might include creating references from web pages delivered over HTTP/TLS; DNS resource records signed using DNSSEC or data values embedded in certificates, Certificate Revocation Lists (CRLs), or other signed data objects.

The Named Identifier can be represented in a number of ways: using the "ni" URI scheme that we specifically define for the name (which is very similar to the "magnet link" that is informally defined in other protocols [magnet]), or using other mechanisms also defined herein. However it is represented, the Named Identifier *\*names\** a resource, and the mechanism used to dereference the name and to *\*locate\** the named resource needs to be known by the entity that dereferences it.

Media content-type, alternative locations for retrieval and other additional information about a resource named using this scheme can be provided using a query string. A companion specification [I-D.hallambaker-decade-ni-params] describes specific values that can be used in such query strings for these various purposes and other extensions to this basic format specification.

In addition, we also define a ".well-known" URL equivalent, and a way to include a hash as a segment of an HTTP URL, as well as a binary format for use in protocols that require more compact names and a human-speakable text form that could be used, e.g., for reading out (parts of) the name over a voice connection.

Not all uses of these names require use of the full hash output - truncated hashes can be safely used in some environments. For this reason, we define a new IANA registry for hash functions to be used with this specification so as not to mix strong and weak (truncated) hash algorithms in other protocol registries.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Syntax definitions in this memo are specified according to ABNF [RFC5234].

## 2. Hashes are what Count

This section contains basic considerations related to how we use hash function outputs that are common to all formats.

When comparing two names of the form defined here, an implementation MUST only consider the digest algorithm and the digest value, i.e.,

it MUST NOT consider other fields defined below (such as an authority field from a URI or any parameters). Implementations MUST consider two hashes identical, regardless of encoding, if the decoded hashes are based on the same algorithm and have the same length and the same binary value. In that case, the two names can be treated as referring to the same thing.

The sha-256 algorithm as specified in [SHA-256] is mandatory to implement, that is, implementations MUST be able to generate/send and to accept/process names based on a sha-256 hash. However implementations MAY support additional hash algorithms and MAY use those for specific names, for example in a constrained environment where sha-256 is non-optimal or where truncated names are needed to fit into corresponding protocols (when a higher collision probability can be tolerated).

Truncated hashes MAY be supported. When a hash value is truncated the name MUST indicate this. Therefore we use different hash algorithm strings for these, such as sha-256-32 for a 32-bit truncation of a sha-256 output. A 32-bit truncated hash is essentially useless for security in almost all cases, but might be useful for naming. With current best practices [RFC3766] very few, if any, applications making use of names with less than 100 bit long hashes will have useful security properties.

When a hash value is truncated to N bits the left-most N bits, that is, the most significant N bits in network byte order, from the binary representation of the hash value MUST be used as the truncated value. An example of a 128-bit hash output truncated to 32 bits is shown in Figure 2.

```
128-bit hash: 0x265357902felb7e2a04b897c6025d7a2
32-bit truncated hash: 0x26535790
```

Figure 2: Example of Truncated Hash

When the input to the hash algorithm is a public key value, as may be used by various security protocols, the hash SHOULD be calculated over the public key in an X.509 SubjectPublicKeyInfo structure (Section 4.1 of [RFC5280]). This input has been chosen primarily for compatibility with DANE [I-D.ietf-dane-protocol], but also includes any relevant public key parameters in the hash input, which is sometimes necessary for security reasons. This does not force use of X.509 or full compliance with [RFC5280] since formatting any public key as a SubjectPublicKeyInfo is relatively straightforward and well supported by libraries.

Any of the formats defined below can be used to represent the resulting name for a public key.

Other than in the above special case where public keys are used, we do not specify the hash function input here. Other specifications are expected to define this.

### 3. Named Information (ni) URI Format

A Named Information (ni) URI consists of the following nine components:

**Scheme Name** The scheme name is 'ni'.

**Colon and Slashes** The literal "://"

**Authority** The optional authority component may assist applications in accessing the object named by an ni URI. There is no default value for the authority field. (See [RFC3986] Section 3.2.2 for details.) While ni names with and without an authority differ syntactically from ni names with different authorities, all three refer to the same object if and only if the digest algorithm, length, and value are the same.

**One slash** The literal "/"

**Digest Algorithm** The name of the digest algorithm, as specified in the IANA registry defined in Section 9.4 below.

**Separator** The literal ";"

**Digest Value** The digest value MUST be encoded using the base64url [RFC4648] encoding, with no "=" padding characters.

**Query Parameter separator '?'** The query parameter separator acts as a separator between the digest value and the query parameters (if specified). For compatibility with IRIs, non-ASCII characters in the query part MUST be encoded as UTF-8, and the resulting octets MUST be %-encoded (see [RFC3986] Section 2.1).

**Query Parameters** A tag=value list of optional query parameters as are used with HTTP URLs [RFC2616] with a separator character '&' between each. For example, "foo=bar&baz=bat"

It is OPTIONAL for implementations to check the integrity of the URI/resource mapping when sending, receiving or processing "ni" URIs.

Escaping of characters follows the rules in RFC 3986. This means that %-encoding is used to distinguish between reserved and unreserved functions of the same character in the same URI component. As an example, an ampersand ('&') is used in the query part to separate attribute-value pairs; an ampersand in a value therefore has to be escaped as '%26'. Note that the set of reserved characters differs for each component, as an example, a slash ('/') does not have any reserved function in a query part and therefore does not have to be escaped. However, it can still appear escaped as '%2f' or '%2F', and implementations have to be able to understand such escaped forms. Also note that any characters outside those allowed in the respective URI component have to be escaped.

The Named Information URI adapts the URI definition from the URI Generic Syntax [RFC3986]. We start with the base URI production:

```
URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ]  
    ; from RFC 3986
```

Figure 3: URI syntax

Adapting that for the Named Information URI:

```
NI-URI      = ni-scheme ":" ni-hier-part [ "?" query ]  
              ; adapted from "URI" in RFC 3986  
              ; query is from RFC 3986, Section 3.4  
ni-scheme    = "ni"  
ni-hier-part = "://" [ authority ] "/" alg-val  
              ; authority is from RFC 3986, Section 3.2  
alg-val      = alg ";" val  
              ; adapted from "hier-part" in RFC 3986  
alg          = 1*unreserved  
val          = 1*unreserved  
              ; unreserved is from RFC 3986, Section 2.3
```

Figure 4: ni Name syntax

The "val" field MUST contain the output of base64url encoding (with no "=" padding characters) the result of applying the hash function ("alg") to its defined input, which defaults to the object bytes that are expected to be returned when the URI is dereferenced.

Relative ni URIs can occur. In such cases, the algorithm in [RFC3986] Section 5 applies. As an example, in Figure 5, the absolute URI for "this third document" is "ni://example.com/sha-256-128;...".

```
<html>
  <head>
    <title>ni: relative URI test</title>
    <base href="ni://example.com">
  </head>

  <body>
    <p>Please check <a href="sha-256;f40xZX...">this document</a>.
      and <a href="sha-256;UyaQV...">this other document</a>.
      and <a href="sha-256-128;...">this third document</a>.
    </p>
  </body>
</html>
```

Figure 5: Example HTML with relative ni URI

The authority field in an ni URI is not quite the same as that from an HTTP URL, even though the same values (e.g., DNS names) may be usefully used in both. For an ni URI, the authority does not control nearly as much of the structure of the "right hand side" of the URI. With ni URIs we also define standard query string attributes and of course have a strictly defined way to include the hash value.

Internationalisation of strings within ni names is handled exactly as for http URIs - see [I-D.ietf-httpbis-pl-messaging] Section 2.7.

### 3.1. Content Type Query String Attribute

The semantics of a digest being used to establish a secure reference from an authenticated source to an external source may be a function of associated meta data such as the content type. The Content Type "ct" parameter specifies the MIME Content Type of the associated data as defined in [I-D.ietf-appsawg-media-type-regs]. See Section 9.5 for the associated IANA registry for ni parameter names. as shown in Figure 6. Implementations of this specification MUST support parsing the ct= query string attribute name.

ni:///sha-256-32;f40xZQ?ct=text/plain

Figure 6: Example ni URI with Content Type

Protocols making use of ni URIs will need to specify how to verify name-data integrity for the MIME Content Types that they need to process and will need to take into account possible Content-Transfer-Encoding and other aspects of MIME encoding.

Implementations of this specification SHOULD support name-data



integrity validation for at least the application/octet-stream Content Type with no explicit Content-Transfer-Encoding (which is equivalent to binary). Additional Content Types and Content-Transfer-Encodings can of course also be supported, but are OPTIONAL. Note that the hash is calculated after the Content Transfer Encoding is removed, so it is applied to the raw data.

If a) the user agent is sensitive to the Content Type and b) the ni name used has a ct= query string attribute and c) the object is retrieved (from a server) using a protocol that specifies a Content Type, then, if the two Content Types match, all is well. If, in this situation, the Content Types do not match, then the client SHOULD handle that situation as a potential security error. Content Type matching rules are defined in [RFC2045] Section 5.1.

#### 4. .well-known URI

We define a mapping between URIs following the ni URI scheme and HTTP [RFC2616] or HTTPS [RFC2818] URLs that makes use of the .well-known URI [RFC5785] by defining an "ni" suffix (see Section 9).

The HTTP(S) mapping MAY be used in any context where clients with support for ni URIs are not available.

Since the .well-known name-space is not intended for general information retrieval, if an application de-references a .well-known/ni URL via HTTP(S), then it will often receive a 3xx HTTP redirection response. A server responding to a request for a .well-known/ni URL will often therefore return a 3xx response and a client sending such a request MUST be able to handle that, as should any fully compliant HTTP [RFC2616] client.

For an ni name of the form "ni://n-authority/alg/val?query-string" the corresponding HTTP(S) URL produced by this mapping is "http://h-authority/.well-known/ni/alg/val?query-string", where "h-authority" is derived as follows: If the ni name has a specified authority (i.e., the n-authority is non-empty) then the h-authority MUST have the same value. If the ni name has no authority specified (i.e., the n-authority string is empty), a h-authority value MAY be derived from the application context. For example, if the mapping is being done in the context of a web page then the origin [RFC6454] for that web site can be used. Of course, there are in general no guarantees that the object named by the ni URI will be available via the corresponding HTTP(S) URL. But in the case that any data is returned, the retriever can determine whether or not it is content that matches the ni URI.

If an application is presented with a HTTP(S) URL with `"/.well-known/ni/"` as the start of its pathname component, then the reverse mapping to an ni URI either including or excluding the authority might produce an ni URI that is meaningful, but there is no guarantee that this will be the case.

When mapping from an ni URI to a .well-known URL, an implementation will have to decide between choosing an "http" or "https" URL. If the object referenced does in fact match the hash in the URL, then there is arguably no need for additional data integrity, if the ni URI or .well-known URL was received "securely." However TLS also provides confidentiality, so there can still be reasons to use the "https" URL scheme even in this case. Additionally, web server policy such as [I-D.ietf-websec-strict-transport-sec] may dictate that data might only be available over "https". In general however, whether to use "http" or "https" is something that needs to be decided by the application.

## 5. URL Segment Format

Some applications may benefit from using hashes in existing HTTP URLs or other URLs. To do this one simply uses the "alg-val" production from the ni name scheme ABNF which may be included for example in the pathname, query string or even fragment components of HTTP URLs [RFC2616]. In such cases there is nothing present in the URL that ensures that a client can depend on compliance with this specification, so clients MUST NOT assume that any URL with a pathname component that matches the "alg-val" production was in fact produced as a result of this specification. That URL might or might not be related to this specification, only the context will tell.

## 6. Binary Format

If a more space-efficient version of the name is needed, the following binary format can be used. The binary format name consists of two fields: a header and the hash value. The header field defines how the identifier has been created and the hash value contains a (possibly truncated) result of a one-way hash over whatever is being identified by the hash value. The binary format of a name is shown in Figure 7.

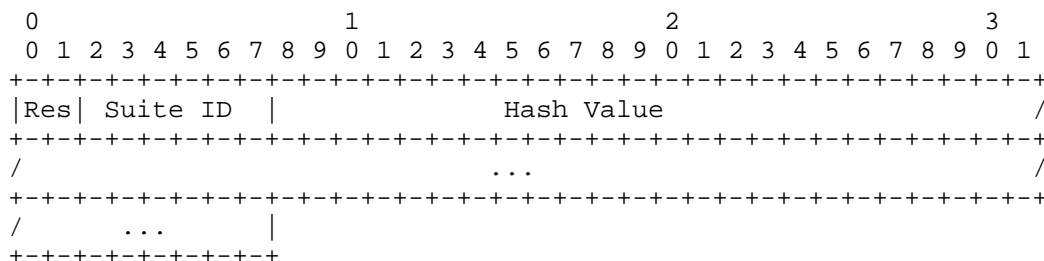


Figure 7: Binary Name Format

The Res field is a reserved 2-bit field for future use and MUST be set to zero for this specification and ignored on receipt.

The hash algorithm and truncation length are specified by the Suite ID. For maintaining efficient encoding for the binary format, only a few hash algorithms and truncation lengths are supported. See Section 9.4 for details.

A hash value that is truncated to 120 bits will result in the overall name being a 128-bit value which may be useful for protocols that can easily use 128-bit identifiers.

## 7. Human-speakable (nih) URI Format

Sometimes a resource may need to be referred to via a name in a format that is easy for humans to read out, and less likely to be ambiguous when heard. This is intended to be usable for example over the phone in order to confirm the (current or future) presence or absence of a resource. This "confirmation" use-case described further in Section 8.3 is the main current use-case for nih URIs.

The ni URI format is not well-suited for this, as, for example, base64url uses both upper and lower case which can easily cause confusion. For this particular purpose, ("speaking" the value of a hash output) the more verbose but less ambiguous (when spoken) nih URI scheme is defined. "nih" stands for "Named Information for Humans." (Or possibly "Not Invented Here," which is clearly false, and therefore worth including [RFC5513]:-)

The justification for nih being a URI scheme is that that can help a user agent for the speaker to better display the value, or help a machine to better speak or recognise the value when spoken. We do not include the query string since there is no way to ensure that its value might be spoken unambiguously, and similarly for the authority, where e.g., some internationalised forms of domain name might not be

easy to speak and comprehend easily. This leaves the hash value as the only part of the ni URI that we feel can be usefully included. But since speakers or listeners (or speech recognition) may err, we also include a check-digit to catch common errors, and allow for the inclusion of "-" separators to make nih URIs more easy to read out.

Fields in nih URIs are separated by a semi-colon (;) character. The first field is a hash algorithm string, as in the ni URI format. The hash value is represented using lower-case ASCII hex characters, for example an octet with the decimal value 58 (0x3A) is encoded as '3a'. This is the same as base16 encoding as defined in RFC 4648 [RFC4648] except using lower-case letters. Separators ("-") characters) MAY be interspersed in the hash value in any way to make those easier to read, typically grouping four or six characters with a separator between.

The hash value MAY be followed by a semi-colon ';' then a checkdigit. The checkdigit MUST be calculated using Luhn's mod N algorithm (with N=16) as defined in [ISOIEC7812], (see also [http://en.wikipedia.org/wiki/Luhn\\_mod\\_N\\_algorithm](http://en.wikipedia.org/wiki/Luhn_mod_N_algorithm)). The input to the calculation is the ASCII-HEX encoded hash value (i.e., "sepval" in the ABNF production below) but with all "-" separator characters first stripped out. This maps the ASCII-HEX so that '0'=0,...'9'=9,'a'=10,...'f'=15. None of the other fields, nor any "-" separators, are input when calculating the checkdigit.

```
humanname = "nih:" alg-sepval [ ";" checkdigit ]
alg-sepval = alg ";" sepval
sepval     = 1*(ahlc / "-")
ahlc       = DIGIT / "a" / "b" / "c" / "d" / "e" / "f"
           ; DIGIT is defined in RFC 5234 and is 0-9
checkdigit = ahlc
```

Figure 8: Human-speakable syntax

For algorithms that have a Suite ID reserved (see Figure 11), the alg field MAY contain the ID value as a ASCII encoded decimal number instead of the hash name string (for example, "3" instead of "sha-256-120"). Implementations MUST be able to match the decimal ID values for the algorithms and hash lengths that they support even if they do not support the binary format.

There is no such thing as a relative nih URI.

## 8. Examples

### 8.1. Hello World!

The following ni URI is generated from the text "Hello World!" (without the quotes, being 12 characters), using the sha-256 algorithm shown with and without an authority field:

```
ni:///sha-256;f40xZX_x_F05LcGBSKHWXfwtSx-jlncoSt3SABJtkGk
```

```
ni://example.com/sha-256;f40xZX_x_F05LcGBSKHWXfwtSx-jlncoSt3SABJtkGk
```

The following HTTP URL represents a mapping from the previous ni name based on the algorithm outlined above.

```
http://example.com/.well-known/ni/sha-256/
f40xZX_x_F05LcGBSKHWXfwtSx-jlncoSt3SABJtkGk
```

### 8.2. Public Key Examples

Given the DER-encoded SubjectPublicKeyInfo in Figure 9 we derive the names shown in Figure 10 for this value.

```

00000000 30 82 01 22 30 0d 06 09 2a 86 48 86 f7 0d 01 01
00000020 01 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01
00000040 00 a2 5f 83 da 9b d9 f1 7a 3a 36 67 ba fd 5a 94
00000060 0e cf 16 d5 5a 55 3a 5e d4 03 b1 65 8e 6d cf a3
00000100 b7 db a4 e7 cc 0f 52 c6 7d 35 1d c4 68 c2 bd 7b
00000120 9d db e4 0a d7 10 cd f9 53 20 ee 0d d7 56 6e 5b
00000140 7a ae 2c 5f 83 0a 19 3c 72 58 96 d6 86 e8 0e e6
00000160 94 eb 5c f2 90 3e f3 a8 8a 88 56 b6 cd 36 38 76
00000200 22 97 b1 6b 3c 9c 07 f3 4f 97 08 a1 bc 29 38 9b
00000220 81 06 2b 74 60 38 7a 93 2f 39 be 12 34 09 6e 0b
00000240 57 10 b7 a3 7b f2 c6 ee d6 c1 e5 ec ae c5 9c 83
00000260 14 f4 6b 58 e2 de f2 ff c9 77 07 e3 f3 4c 97 cf
00000300 1a 28 9e 38 a1 b3 93 41 75 a1 a4 76 3f 4d 78 d7
00000320 44 d6 1a e3 ce e2 5d c5 78 4c b5 31 22 2e c7 4b
00000340 8c 6f 56 78 5c a1 c4 c0 1d ca e5 b9 44 d7 e9 90
00000360 9c bc ee b0 a2 b1 dc da 6d a0 0f f6 ad 1e 2c 12
00000400 a2 a7 66 60 3e 36 d4 91 41 c2 f2 e7 69 39 2c 9d
00000420 d2 df b5 a3 44 95 48 7c 87 64 89 dd bf 05 01 ee
00000440 dd 02 03 01 00 01

00000000 53 26 90 57 e1 2f e2 b7 4b a0 7c 89 25 60 a2 d7
00000020 53 87 7e b6 2f f4 4d 5a 19 00 25 30 ed 97 ff e4

```

Figure 9: A SubjectPublicKeyInfo used in examples and its sha-256 hash

URI: ni:///sha-256;UyaQV-Ev4rdLoHyJJWCi1lOHfrYv9ElaGQAlMO2X_-Q
.well-known URL (split over 2 lines): http://example.com/.well-known/ni/sha256/ UyaQV-Ev4rdLoHyJJWCi1lOHfrYv9ElaGQAlMO2X_-Q
URL Segment: sha-256;UyaQV-Ev4rdLoHyJJWCi1lOHfrYv9ElaGQAlMO2X_-Q
Binary name (ASCII hex encoded) with 120-bit truncated hash value which is Suite ID 0x03: 0353 2690 57e1 2fe2 b74b a07c 8925 60a2
Human-speakable form of a name for this key (truncated to 120 bits in length) with checkdigit: nih:sha-256-120;5326-9057-e12f-e2b7-4ba0-7c89-2560-a2;f
Human-speakable form of a name for this key (truncated to 32 bits in length) with checkdigit and no "-" separators: nih:sha-256-32;53269057;b
Human-speakable form using decimal presentation of the algorithm ID (sha-256-120) with checkdigit: nih:3;532690-57e12f-e2b74b-a07c89-2560a2;f

Figure 10: Example Names

### 8.3. nih Usage Example

Alice has set up a server node with an RSA key pair. She uses an ni URI as the name for the public key that corresponds to the private key on that box. Alice's node might identify itself using that ni URI in some protocol.

Bob would like to believe that its really Alice's node when his node interacts with the network and asks his friend Alice to tell him what public key she uses. Alice hits the "tell someone the name of the public key" button on her admin user interface, and that displays the nih URI and says "tell this to your buddy." She phones Bob and reads the nih URI to him.

Bob types that in to his "manage known nodes" admin application, (or lets that application listen to part of the call), which can regenerate the ni URI and store that or some equivalent. Then when Bob's node interacts with Alice's node it can more safely accept a

signature or encrypt data to Alice's node.

## 9. IANA Considerations

### 9.1. Assignment of ni URI Scheme

The procedures for registration of a URI scheme are specified in RFC 4395 [RFC4395]. The following is the proposed assignment template.

URI scheme name: ni

Status: Permanent

URI scheme syntax. See Section 3

URI scheme semantics. See Section 3

Encoding considerations. See Section 3

Applications/protocols that use this URI scheme name: General applicability.

Interoperability considerations: Defined here.

Security considerations: See Section 10

Contact: Stephen Farrell, [stephen.farrell@cs.tcd.ie](mailto:stephen.farrell@cs.tcd.ie)

Author/Change controller: IETF

References: As specified in this document

### 9.2. Assignment of nih URI Scheme

The procedures for registration of a URI scheme are specified in RFC 4395 [RFC4395]. The following is the proposed assignment template.

URI scheme name: nih

Status: Permanent

URI scheme syntax. See Section 7

URI scheme semantics. See Section 7

Encoding considerations. See Section 7

Applications/protocols that use this URI scheme name: General applicability.

Interoperability considerations: Defined here.

Security considerations: See Section 10

Contact: Stephen Farrell, [stephen.farrell@cs.tcd.ie](mailto:stephen.farrell@cs.tcd.ie)

Author/Change controller: IETF

References: As specified in this document

### 9.3. Assignment of .well-known 'ni' URI

The procedures for registration of a Well Known URI entry are specified in RFC 5785 [RFC5785]. The following is the proposed assignment template.

URI suffix: ni

Change controller: IETF

Specification document(s): This document

Related information: None

### 9.4. Creation of Named Information Hash Algorithm Registry

IANA is requested to create a new registry for hash algorithms as used in the name formats specified here and called the "Named Information Hash Algorithm Registry". Future assignments are to be made through Expert Review [RFC5226]. This registry has five fields, the suite ID, the hash algorithm name string, the truncation length, the underlying algorithm reference and a status field that indicates if algorithm is current or deprecated and should no longer be used. The status field can have the value "current" or "deprecated". Other values are reserved for possible future definition.

If the status is "current", then that does not necessarily mean that the algorithm is "good" for any particular purpose, since the cryptographic strength requirements will be set by other applications or protocols.

A request to mark an entry as "deprecated" can be done by sending a mail to the Designated Expert. Before approving the request, the community MUST be consulted via a "call for comments" of at least two weeks by sending a mail to the IETF discussion list.



Initial values are specified below. The Designated Expert SHOULD generally approve additions that reference hash algorithms that are widely used in other IETF protocols. In addition, the Designated Expert SHOULD NOT accept additions where the underlying hash function (with no truncation) is considered weak for collisions. Part of the reasoning behind this last point is that inclusion of code for weak hash functions, e.g. the MD5 algorithm, can trigger costly false-positives if code is audited for inclusion of obsolete ciphers. See for example [RFC6149],[RFC6150] and [RFC6151] for some hash functions that are considered obsolete in this sense.

The suite ID field ("ID") can be empty, or can have values between 0 and 63, inclusive. Because there are only 64 possible values, this field is OPTIONAL (leaving it empty if omitted). Where the binary format is not expected to be used for a given hash algorithm, this field SHOULD be omitted. If an entry is registered without a suite ID, the Designated Expert MAY allow for later allocation of a suite ID, if that appears warranted. The Designated Expert MAY consult the community via a "call for comments" by sending a mail to the IETF discussion list before allocating a suite ID.

ID	Hash name string	Value length	Reference	Status
0	Reserved			
1	sha-256	256 bits	[SHA-256]	current
2	sha-256-128	128 bits	[SHA-256]	current
3	sha-256-120	120 bits	[SHA-256]	current
4	sha-256-96	96 bits	[SHA-256]	current
5	sha-256-64	64 bits	[SHA-256]	current
6	sha-256-32	32 bits	[SHA-256]	current
32	Reserved			

Figure 11: Suite Identifiers

The Suite ID value 32 is reserved for compatibility with ORCHIDS [RFC4843].

The referenced hash algorithm matching to the Suite ID, truncated to the length indicated, according to the description given in Section 2, is used for generating the hash. The Designated Expert is responsible for ensuring that the document referenced for the hash algorithm meets the "specification required" rule."

#### 9.5. Creation of Named Information Parameter Registry

IANA is requested to create a new registry entitled "Named Information URI Parameter Definitions".

The policy for future assignments to the registry is Expert Review, and as for the ni Hash Algorithm Registry above, the Designated Expert is responsible for ensuring that the document referenced for the parameter definition meets the "specification required" rule."

The fields in this registry are the parameter name, a description and a reference. The parameter name MUST be such that it is suitable for use as a query string parameter name in an ni URI. (See Section 3.)

The initial contents of the registry are:

Parameter	Meaning	Reference
-----	-----	-----
ct	Content Type	[RFC-THIS]

## 10. Security Considerations

No secret information is required to generate or verify a name of the form described here. Therefore a name like this can only provide evidence for the integrity for the referenced object and the proof of integrity provided is only as good as the proof of integrity for the name from which we started. In other words, the hash value can provide a name-data integrity binding between the name and the bytes returned when the name is de-referenced using some protocol.

Disclosure of a name value does not necessarily entail disclosure of the referenced object but may enable an attacker to determine the contents of the referenced object by reference to a search engine or other data repository or, for a highly formatted object with little variation, by simply guessing the value and checking if the digest value matches. So the fact that these names contain hashes does not protect the confidentiality of the object that was input to the hash.

The integrity of the referenced content would be compromised if a weak hash function were used. SHA-256 is currently our preferred hash algorithm which is why we've only added SHA-256 based suites to the initial IANA registry.

If a truncated hash value is used, certain security properties will be affected. In general a hash algorithm is designed to produce sufficient bits to prevent a 'birthday attack' collision occurring. To ensure that the difficulty of discovering two pieces of content that result in the same digest with a work factor  $O(2^x)$  by brute force requires a digest length of  $2x$ . Many security applications only require protection against a 2nd pre-image attack which only requires a digest length of  $x$  to achieve the same work factor.

Basically, the shorter the hash value used, the less security benefit you can possibly get.

An important thing to keep in mind is not to make the mistake of thinking two names are the same when they aren't. For example, a name with a 32 bit truncated sha-256 hash is not the same as a name with the full 256 bits of hash output, even if the hash value for one is a prefix of that for the other.

The reason for this is that if an application treats those as the same name then that might open up a number of attacks. For example, if I publish an object with the full hash, then I probably (in general) don't want some other application to treat a name with just the first 32 bits of that as referring to the same thing, since the 32 bit name will have lots of colliding objects. If ni or nih URIs become widely used, there will be many cases where names will occur more than once in application protocols, and it'll be unpredictable which instance of the name would be used for name-data integrity checking, leading to threats. For this reason, we require that the algorithm, length and value all match before we consider two names to be the same.

The fact that an ni URI includes a domain name in the authority field by itself implies nothing about the relationship between the owner of the domain name and any content referenced by that URI. While a name-data integrity service can be provided using ni URIs, that does not in any sense validate the authority part of the name. For example, there is nothing to stop anyone creating an ni URI containing a hash of someone else's content. Application developers MUST NOT assume any relationship between the registrant of the domain name that is part of an ni URI and some matching content just because the ni URI matches that content.

If name-data integrity is successfully validated, and the hash is strong and long enough, then the "web origin" [RFC6454] for the bytes of the named object is really going to be the place from which you got the ni name and not the place from which you got the bytes of the object. This appears to offer a potential benefit if using ni names for, for example, scripts included from a HTML page accessed via server-authenticated https. If name-data integrity is not validated (and it is optional), or fails, then the web origin is, as usual, the place from which the object bytes were received. Applications making use of ni names SHOULD take this into account in their trust models.

Some implementations might mis-handle ni URIs that include non-base64 characters, whitespace or other non-conforming strings and that could lead to erroneously considering names to be the same when they are not. An ni URI that is malformed in such ways MUST NOT be treated as

matching any other ni URI. Implementers need to check the behaviour of libraries for such parsing problems.

## 11. Acknowledgments

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.

The authors would also like to thank Carsten Bormann, Martin Durst, Tobias Heer, Bjoern Hoehrmann, Tero Kivinen, Barry Leiba, Larry Masinter, David McGrew, Alexey Melnikov, Bob Moskowitz, Jonathan Rees, Eric Rescorla, Zach Shelby, Martin Thomas, for their comments and input to the document. Thanks, in particular, to James Manger for correcting the examples.

## 12. References

### 12.1. Normative References

- [I-D.ietf-appsawg-media-type-regs]  
Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures",  
draft-ietf-appsawg-media-type-regs-14 (work in progress),  
June 2012.
- [I-D.ietf-httpbis-pl-messaging]  
Fielding, R., Lafon, Y., and J. Reschke, "HTTP/1.1, part  
1: Message Routing and Syntax",  
draft-ietf-httpbis-pl-messaging-20 (work in progress),  
July 2012.
- [ISOIEC7812]  
ISO, "ISO/IEC 7812-1:2006 Identification cards --  
Identification of issuers -- Part 1: Numbering system",  
October 2006, <[http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=39698](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39698)>.
- [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.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,

- Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.
- [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.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.
- [SHA-256] NIST, "United States National Institute of Standards and Technology (NIST), FIPS 180-3: Secure Hash Standard", October 2008, <[http://csrc.nist.gov/publications/fips/fips180-3/fips180-3\\_final.pdf](http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf)>.

## 12.2. Informative References

- [I-D.hallambaker-decade-ni-params]  
Hallam-Baker, P., Stradling, R., Farrell, S., Kutscher, D., and B. Ohlman, "The Named Information (ni) URI Scheme: Optional Features", draft-hallambaker-decade-ni-params-03 (work in progress), June 2012.
- [I-D.ietf-dane-protocol]  
Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", draft-ietf-dane-protocol-23 (work in progress), June 2012.
- [I-D.ietf-websec-strict-transport-sec]

- Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", draft-ietf-websec-strict-transport-sec-11 (work in progress), July 2012.
- [RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.
- [RFC4843] Nikander, P., Laganier, J., and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", RFC 4843, April 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, April 1 2009.
- [RFC6149] Turner, S. and L. Chen, "MD2 to Historic Status", RFC 6149, March 2011.
- [RFC6150] Turner, S. and L. Chen, "MD4 to Historic Status", RFC 6150, March 2011.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.
- [magnet] Wikipedia article, "Magnet URI Scheme", April 2012, <[http://en.wikipedia.org/wiki/Magnet\\_link](http://en.wikipedia.org/wiki/Magnet_link)>.
- [ref.ccn] Jacobson at al., "Networking Named Content", CoNEXT 2009 , December 2009.
- [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](mailto:stephen.farrell@cs.tcd.ie)

Dirk Kutscher  
NEC  
Kurfuersten-Anlage 36  
Heidelberg,  
Germany

Phone:  
Email: [kutscher@neclab.eu](mailto:kutscher@neclab.eu)

Christian Dannewitz  
University of Paderborn  
Paderborn  
Germany

Email: [cdannewitz@upb.de](mailto:cdannewitz@upb.de)

Borje Ohlman  
Ericsson  
Stockholm S-16480  
Sweden

Email: [Borje.Ohlman@ericsson.com](mailto:Borje.Ohlman@ericsson.com)

Ari Keranen  
Ericsson  
Jorvas 02420  
Finland

Email: [ari.keranen@ericsson.com](mailto:ari.keranen@ericsson.com)

Phillip Hallam-Baker  
Comodo Group Inc.

Email: [philliph@comodo.com](mailto:philliph@comodo.com)





ICN Research Group  
Internet-Draft  
Intended status: Informational  
Expires: January 10, 2013

L. Li  
X. Xu  
J. Wang  
Z. Hao  
ZTE Corporation  
July 9, 2012

Information-Centric Network in an ISP  
draft-li-icnrg-icn-isp-00

Abstract

ICN (Information-Centric Network) may be deployed over different underlying networks, e.g. ad hoc networks, DTN and ISP's networks. This document discusses deploying ICN in ISP's existing networks and ICN design for ISP.

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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Deployment Considerations in an ISP . . . . .	3
4. Routing and Caching Control . . . . .	4
5. ICN for ISP Example . . . . .	5
6. Security Considerations . . . . .	6
7. References . . . . .	6
7.1. Normative References . . . . .	6
7.2. Informative References . . . . .	7
Authors' Addresses . . . . .	7

## 1. Introduction

ICN (Information-Centric Network) may be deployed over different underlying networks, e.g. ad hoc networks, DTN and ISP's networks. This document discusses deploying ICN in ISP's existing networks and ICN design for ISP.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 3. Deployment Considerations in an ISP

Information-centric networks can be deployed on top of layer-3 or layer-2 networks. It should be preferable for ISPs to deploy ICN as an overlay network on top of layer-3 networks, for the following considerations: firstly, in the case of incremental deployment, packets between newly deployed content routers have to go through ordinary routers which do not understand ICN protocols; secondly, content routers should be preferably deployed in areas with requirements of reducing cost or improving Quality of Service (QoS), and there is no necessity of deployment in areas where QoS requirements can be fulfilled, and link cost is lower.

Content routers may be deployed at the edges of networks close to content consumers, for the following considerations: firstly, early cache hit at network edges means better QoS and more link cost savings; secondly, deploying caches at network edges can mitigate the impact of unstable wireless link in the case of mobile access users; thirdly, it is easier to handle the requests since traffic is light at network edges, and cache hits at network edges reduce the load at content routers in core network which forwarding high volume traffic.

Content routers with huge cache spaces may be deployed in core networks to achieve high cache hit rates. Research on cache, e.g. [web\_caching] and [cooperative\_caching], shows that both cache size and serving user number affect cache hit rate. Though early cache hit is better, cache hit rate at network edge is limited. An edge content router's cache hit rate is limited by its cache size and serving user number. Firstly, in order to reach a high cache hit rate, huge cache space is needed. But it's costly to deploy huge cache spaces in large number of edge content routers. Secondly, fewer users are served by an edge content router. As a result, a large proportion of content requests are for one-time access

contents, and hit rate is limited at network edges.

It is not necessary to deploy a deep hierarchy of content routers in an ISP. On one hand, it is easier to deploy fewer content routers in current network. On the other hand, it is preferable that the cache space of a content router is much bigger than the one in a lower tier, which means the number of tiers is small. Because of the Zipf-like distribution of content requests, the cache size must grow exponentially when the tier grows. Otherwise, cache hit rate of each non-bottom tier is very low.

#### 4. Routing and Caching Control

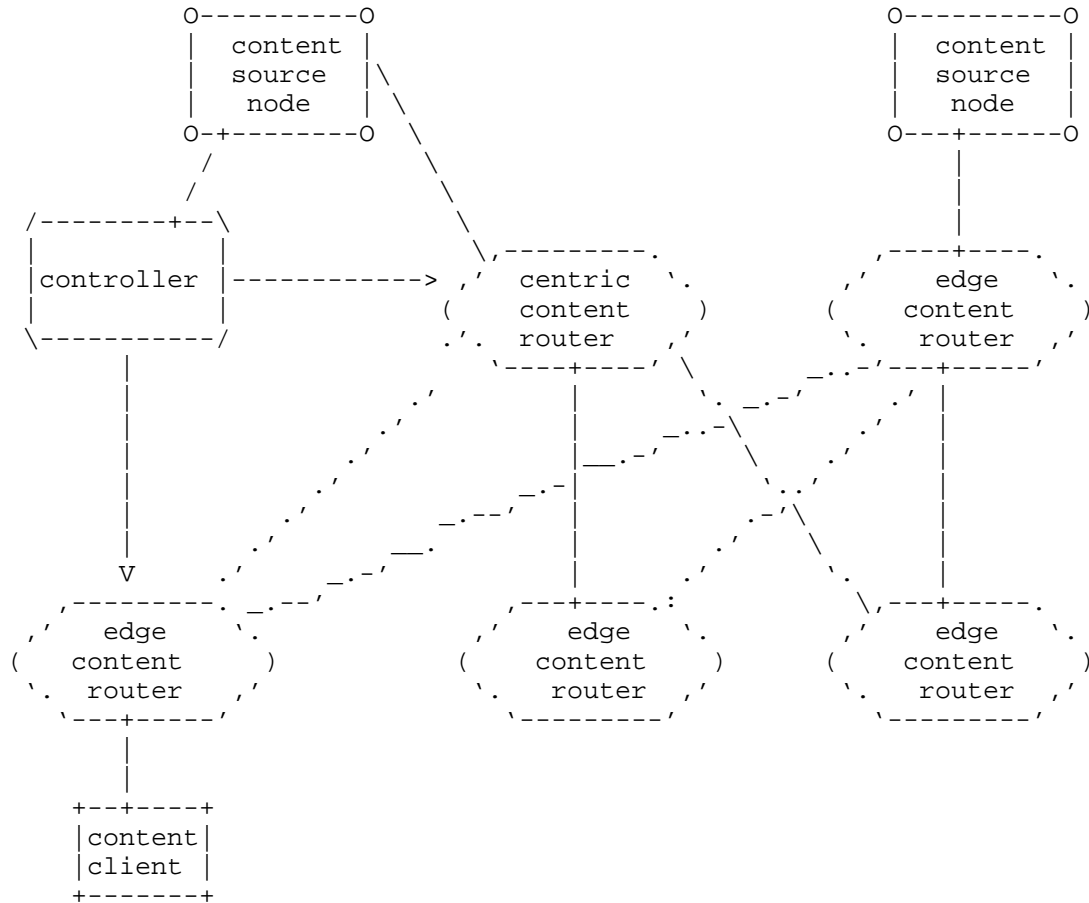
There are two ways to collect topology data and generate routing table, self-generation and centralized generation. In the self-generation way, content routers run a routing protocol to exchange topology data inside an AS or among ASes. Then each content router runs a routing algorithm locally to generate a routing table. Alternatively, inside an AS, content routing tables can be generated by the centralized way. In this way, one or more controllers collect topology data, and generate routing tables for content routers. Then the controller(s) sends route entries to content routers.

There are also two ways to control caching. A content router can decide to cache a content or not on its own by running a cache replacement algorithm like LRU or LFU. However, an ISP may also want to use centralized controller(s) to enforce some cache policies.

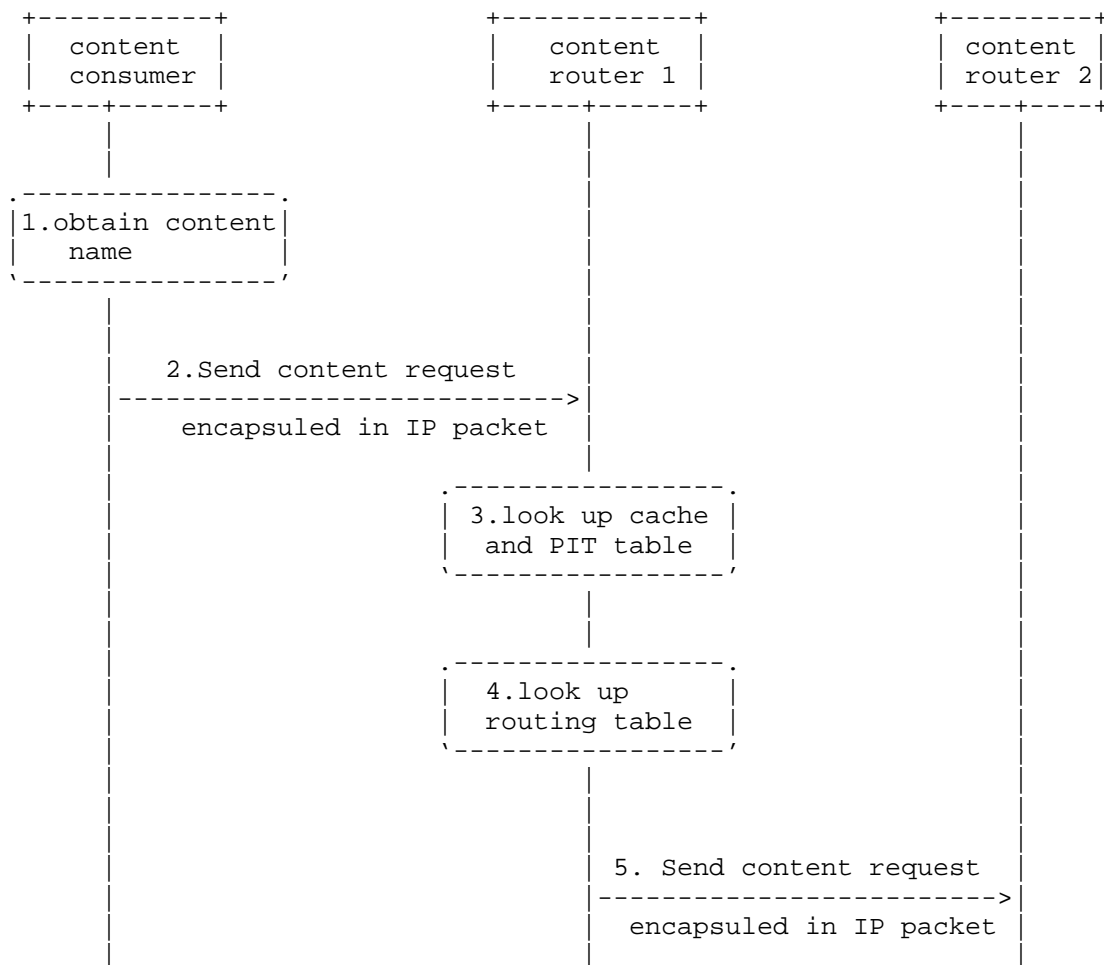
An ISP may utilize centralized controller(s) to enforce routing and cache policy under following considerations. First, to meet QoS requirement, an ISP may decide routing path and cache resource assignment based on factors like content type, content download frequency and distance to content source. Second, to reduce link cost, an ISP may assign more cache resource for the contents passing through costly links by controlling routing path and/or cache priority. Third, to balance link load and cache load, an ISP may optimize routes based on load status. Fourth, an ISP may provide better services to paid users or content providers by controlling routing path and/or cache priority.

To control routing and caching, an ICN controller may need to collect not only topology data and traffic data, but also content data like content type and content download frequency.

## 5. ICN for ISP Example



The above figure shows an example of ICN in an ISP. In this example, there are two tiers of content routers, a tier of edge content routers with small cache spaces and a tier of centric content routers with huge cache spaces. To store massive contents, the centric content routers use cache clusters. The ICN network is an overlay network deployed over IP network. An ICN controller is responsible for generating routing tables and sending route entries to content routers. Content request routing in the example is shown in the figure below.



## 6. Security Considerations

TBD

## 7. References

### 7.1. Normative References

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

## 7.2. Informative References

## [cooperative\_caching]

Wolman, A., Voelker, G., Sharma, N., Cardwell, N., Karlin, A., and H. Levy, "On the Scale and Performance of Cooperative Web Proxy Caching", ACM Symposium on Operating Systems Principles, 1999.

## [web\_caching]

Breslau, L., Cue, P., Cao, P., Fan, L., Phillips, G., and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", INFOCOM, 1999.

## Authors' Addresses

Lichun Li  
ZTE Corporation  
RD Building 1,Zijinghua Road No.68  
Yuhuatai District,Nanjing 210012  
P.R.China

Email: li.lichun1@zte.com.cn

Xin Xu  
ZTE Corporation  
RD Building 1,Zijinghua Road No.68  
Yuhuatai District,Nanjing 210012  
P.R.China

Email: xu.xin18@zte.com.cn

Jun Wang  
ZTE Corporation  
RD Building 1,Zijinghua Road No.68  
Yuhuatai District,Nanjing 210012  
P.R.China

Email: wang.jun17@zte.com.cn



Zhenwu Hao  
ZTE Corporation  
RD Building 1, Zijinghua Road No.68  
Yuhuatai District, Nanjing 210012,  
P.R.China

Email: hao.zhenwu@zte.com.cn



Internet Engineering Task Force  
Internet-Draft  
Expires: December 24, 2012

P. Montessoro  
R. Bernardini  
UniUD  
June 22, 2012

REBOOK: a Network Resource Booking Algorithm  
draft-montessoro-rebook-00

Abstract

This document describes REBOOK, a resource reservation algorithm for deterministic and fast dynamic resource allocation and release. Based on a stateful approach, it handles faults and network errors, and recovers from route changes and unexpected flows shutdown. The distributed scheme used to store flows information have guaranteed constant complexity.

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 December 24, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	3
2. REBOOK Description . . . . .	6
2.1. Message transport and format . . . . .	6
2.2. Node behaviour . . . . .	7
2.2.1. Start-up . . . . .	7
2.2.2. Keep-Alive . . . . .	8
2.2.3. Data transmission . . . . .	9
2.2.4. Reservation update . . . . .	9
2.2.5. Path change . . . . .	9
2.2.6. Shutdown . . . . .	10
2.3. Commands . . . . .	10
2.4. Router behaviour . . . . .	11
2.4.1. On reception of a RESV request . . . . .	11
2.4.2. On reception of a data packet with the reference field . . . . .	12
2.4.3. On reception of a KPALV request . . . . .	12
2.4.4. On reception of a UP_RESV request . . . . .	13
2.4.5. On reception of a RL_RESV request . . . . .	13
2.4.6. Checking packet validity . . . . .	13
3. Using the IP Router Alert field . . . . .	14
4. Session Timeout . . . . .	14
5. What is a resource? . . . . .	15
6. Security consideration . . . . .	15
7. IANA considerations . . . . .	15
8. References . . . . .	15
8.1. References . . . . .	15
8.2. Informative References . . . . .	16

## 1. Introduction

A large number of applications prefer UDP as a transport protocol. Among others, teleconferencing, streaming applications, real-time communications, and networked multiplayer games; more generally, applications whose information value and perceived quality strictly relies on network delay, and also multi-cast services. Dynamics of network load can impact negatively on multimedia applications; however, UDP doesn't offer any native congestion control mechanism and therefore it cannot guarantee Quality of Service (QoS). In addition, congestion is often caused by these applications' high bandwidth requirements.

On the other side, TCP provides a congestion control mechanism based on message loss only, without any explicit knowledge about the traffic load in intermediate network nodes along the communication path. Therefore, both UDP and TCP sources need to be enhanced with a congestion control algorithm not only aimed at reducing loss rate and improving bandwidth utilization, but also at improving fairness towards competing connections while satisfying related QoS requirements. UDP flows should become TCP-friendly whereas UDP and TCP flows should both become aware of network load in order to adjust their transmission rate before losing packets.

In the last years, several methods have been introduced for QoS control mechanisms, mainly based on adapting the sender's transmission rate in accordance with the network congestion state (see Sect. 2), but typically with such approaches no QoS guarantees can be made effectively. The proposed algorithm, that we call REBOOK, allows a control protocol to prevent congestion by reserving enough resources for supporting a dynamically changing QoS level in advance, therefore it would be the most straightforward approach to manage the problem.

It must be outlined that REBOOK is not dependent on TCP/IP protocols, as it can be used in any other packet switching network. Obviously, in the following the Internet and the TCP/IP protocol suite will be used as reference environment for its description.

REBOOK requires a hosting protocol to carry the algorithm's messages. They can be handled by a dedicated signalling protocol, like RSVP [RFC2205] or a new ad hoc protocol, or it can be embedded in a data transport protocol, like TCP using the options field.

REBOOK is designed to handle virtually every transport-layer connection or application-layer flows, not necessarily in real-time or QoS only. In principle, except for very short flows the network may benefit from the preventive declaration (and the consequent

resource booking) of the amount of traffic that each connection is going to generate: if the application knows the amount of granted resources in advance, it can modulate the data transmission rate to prevent packet loss.

REBOOK implementation will bring to a sender-oriented protocol, unlike RSVP which is receiver oriented. The node initiating a connection is responsible for resource reservation request, based on the amount and type of data to be transmitted, application constraints and, possibly, SLA (Service Level Agreement) parameters. While the connection is active the amount of granted resources may be reduced to allow the activation of new flows in an almost-congested network or may be increased if switching nodes become less loaded. These events are acknowledged by the sender that will consequently adapt its transmission rate. RSVP is receiver-oriented mainly because it is designed to support single-cast and multi-cast flows as well.

REBOOK does not rely on any special network feature: it works even if only part of the network is REBOOK-aware as the resource reservation is effective even if part of a flow traverses unaware routers. There are no special requirements to routing, which can be asymmetrical (transmitting and receiving flows can follow different paths), except, obviously, its stability: in normal conditions data packets and control messages must follow the same route for the duration of the connection. Anyhow, if unexpected events occur, REBOOK identifies route changes and recovers from errors and faults in the network.

REBOOK does not rely on special hardware in routers either. Its status storage scheme allows direct access to table entries without any hardware look-up feature, simply using conventional memory architecture. This makes its implementation faster and cheaper than today's typical solutions provided by hardware hashing. Finally, REBOOK does not require any improvement in the switching fabric. No additional memory for queues and buffers nor different packet handling. On the contrary, the router architecture will drive the resource granting phase, depending on available resources left after previous reservations.

It is worth observing that REBOOK is not another virtual circuit technique; it is an add-on to conventional routers to improve performance and features. For example, it does not require any interaction with the routing protocols; it can work along routes traversing several Autonomous Systems and routers using different routing protocols; it does not require to be supported by all the routers along the path nor for the supporting routers to be contiguous; it does need neither hierarchy nor partitioning. In

respect of existing virtual circuit techniques, the novelty consists in making each router autonomous in handling its REBOOK data structures and the effects that routing dynamics have on them, without any routing-aware signalling protocol. All these features overcome most of the difficulties that limit a wide deployment of MPLS or ATM, for example. Moreover, in case of routing instability or faults, REBOOK can fail in enhancing router performance, but packets are still forwarded in a best-effort, lookup-driven mode.

REBOOK is scalable since its computational cost is constant, regardless of the number of active flows, yielding a solution to per-flow resource reservation and packet handling, without the need of aggregations. Moreover, it is intrinsically secure since each router uses only the information generated in a previous phase by itself, allowing any form of fast and secure symmetrical cryptography without the need of a key exchange.

REBOOK is based on a distributed linked data structure in the sense that it makes use of pointers to memory locations or indexes to table entries containing information stored in the routers that are very likely to be accessed in the future. Unlike conventional linked data structures, pointers are not stored within the router they address, but in the end nodes, in the packets, and in adjacent routers. When a packet is received by a router, the pointer to the table entry to be used for its processing is found in the packet itself and look-up procedures are avoided. To keep the pointers consistent in a dynamic environment, where route changes may send packets to routers different than the ones the pointers refer to, a specific integrity check is adopted. It makes REBOOK completely autonomous in identifying faults, errors and route changes. The overhead introduced by this integrity check is a critical issue. Thanks to careful design of the distributed linked data structure, experimental data demonstrate that this overhead is overcome by the speedup provided by the look-up-free access to table entries.

REBOOK is a soft state algorithm. Reservations no longer refreshed by keep-alive messages (due to errors, route changes or end node faults) are removed by a low priority table cleanup task active in the routers. This is the only part of the algorithm whose cost is linear in the number of active flows, instead of constant. However, the experimental results in [rebook] report 100 ms CPU time to handle a 10,000,000 flows RR table, and in current implementation this procedure is activated every 15 s. Thanks to this procedure and to the consistency check, REBOOK messages are not required to be acknowledged.

Summarizing, the main features of REBOOK are

Soft state: Reservations not refreshed by keep-alive messages are automatically removed.

Scalability: The computational cost for routing and connection handling is constant with the number of active flows. The only cost that grows linearly with the number of active flows is the removal of expired session, but experimental data shows that this is negligible.

Consistency checks: Routers can easily check (in constant time) if the received packet is actually a valid REBOOK-related packet. This makes it possible for routers to identify faults, errors and route changes.

Safety: Each router uses only the information generated in a previous phase by itself, this makes it possible to use efficient cryptographic protection techniques.

## 2. REBOOK Description

In a REBOOK setup there is a `_source node_` that wants to reserve resources to send data to a `_target node_` with a guaranteed quality of service. Before beginning the streaming, the source node reserves the required resources by sending REBOOK requests to the target node. The intermediate REBOOK-capable routers will analyze the REBOOK requests and assign resources to the newly opened connection. This section describes the details of the interaction between the source node, the target node and the intermediate routers.

### 2.1. Message transport and format

It is worth emphasizing that REBOOK is not a protocol, but an algorithm for resource reservations that can be "embedded" in many different protocol. Because of this, we do not specify format for REBOOK requests or how they are transmitted over the network, leaving the choice of those details to the application employing REBOOK. This is similar to what it is done in other protocols like TFRC (TCP-Friendly Rate Control [RFC5348]) that requests that the source node will send some data (e.g., estimated RTT, timestamps, ...) to the receiver and that the receiver will send back some feedback data (e.g., loss probability), but it leaves open how those data are embedded in the protocol employing TFRC.

About the way REBOOK requests are transported, we can observe that there are two possible approaches

- o Via a specific protocol, "parallel" to the one used to stream the data.



- o Piggy-backed to the data stream.

Both solutions are reasonable and the choice will be a matter of convenience of the specific application.

## 2.2. Node behaviour

In this section we describe an overview of the behavior of the different REBOOK nodes (source, target and routers).

### 2.2.1. Start-up

- o Suppose that a source node needs to stream data to the target using a guaranteed QoS. The node decides to reserve the required resources using REBOOK. Toward such a goal, the node sends a REBOOK RESV request (see Section 2.3) to the target node.
- o The packet with the RESV request travels over the network, passing through several routers. If a router is REBOOK-capable, it reserves the requested resources (if available) and update its internal tables by adding an entry for the new stream(see Section 2.4 for details). If not enough resources are available, the router can decide to reserve less than the required resource amount and write this information in the request.
- o In order to be able to recover later the information about the registered stream, the router appends to the RESV packet a \_reference\_ to the new entry.

Although the "reference" written by the router can be thought as the memory address of the entry associated with the stream, it is worth emphasizing that a reference is used only by the router that wrote it, so that, to the other nodes the reference appears as an opaque string of bits. This implies that the reference is not constrained to be an actual address and that the router can give it any meaning without compromising interoperability. For example, the reference could be an \_index\_ to the table, possibly extended with some CRC and encrypted to check for packet validity and improve security (see Section 2.4.6).

- o The RESV request arrives to the target. The RESV packet now contains the amount of resources reserved along the path and the list of table references added by the intermediate REBOOK-capable. The target sends back to the source the received packet with a RESV\_ACK request (see Section 2.3). Now the source knows the amount of reserved resources and the reference list.

### 2.2.2. Keep-Alive

Periodically, starting right after receiving the RESV\_ACK, the source sends keep-alive KPALV requests to the target node. The KPALV request carries the following data

- o The amount of reserved resources
- o The list of table references inserted by the routers

When a router receives a KPALV message, first it checks if it is a valid KPALV request relative to an already opened stream. If the check is positive

1. It updates the amount of reserved resources.

In order to understand why this is necessary, consider the following case: source S wants to send data to target T; suppose that the canonical required bandwidth is 3 Mbit/s, but that transmission can still be done with good enough quality as soon as the bandwidth is at least 512 kbit/s (for example, the source could encode the data at a lower quality). Suppose also that between S and T there are two routers A and B, the latter with only 1 Mbit/s available. When A processes the RESV request, it has enough resources and decide to reserve 3 Mbit/s to S; however, since B has not enough bandwidth it reserves only 1 Mbit/s to S. Note that A is wasting 2 Mbit/s, since S will never be able to use more than 1 Mbit/s, but A reserved 3 Mbit/s to S. However, since the KPALV packet sent by S will contain the amount of reserved resources, A can see that only 1 Mbit/s will be used by S and will update accordingly its tables.

2. It stores in the table entry associated with the stream the reference written by the next router. See Section 2.4 for details about how this value is used.

KPALV messages are expected to be received at regular intervals. If KPALV packets are not received for a time exceeding a threshold (possibly depending on the path RTT), the router will suppose the session expired and will release the reserved resources.

Finally, observe that the target usually ignores the KPALV packets, unless there is a change in the reservation. See Section 2.2.4 for details.

### 2.2.3. Data transmission

In the simplest implementation of REBOOK, no special handling is required for data packets. The routers will access to the routing tables in the usual way.

However, in order to speed-up the access to the table, avoiding the look-up procedure, the source can include in every transmitted data packet the reference value inserted by the first router. (See Section 2.4 for details about how this value is used.) Since this value must be used by the routers, the router must know (i) that the packet belongs to a REBOOK stream and (ii) where the reference is stored. A possible solution, employing the IP Router Alert field [RFC2113] is described in Section 3.

### 2.2.4. Reservation update

In some cases it could be necessary to update the resource reserved. For example, in a video streaming application the user could decide to receive a better quality (but more expensive) version of the video, so the source needs to reserve more resources. Another case where it is necessary to update the reservation is when some routers in the path are not REBOOK capable, so that their service is of best-effort type. If congestion at the non-REBOOK router happens, the source detects it by using known congestion control procedures (e.g., TFRC [RFC5348]) and it can decide to update the reserved bandwidth. Finally, a third case is when a router receives a request that cannot satisfy and it can decide to reduce the amount of resources assigned to another node in order to satisfy the new request.

In all those cases, the reservation can be changed by using a KPALV request with the new reservation. If the source wants to reduce the allocated resources, it just generates the request with the new allocation value; if an intermediate router wants to reduce the reserved bandwidth, it will wait for the first KPALV packet to arrive and it will update the reservation value in it.

When the target receives a KPALV packet with different resource values, it sends to the source a PRL\_ACK (see Section 2.3) with the new values.

Finally, if the source desires to increase the reserved resources, it can use the request UP\_RESV (see Section 2.3).

### 2.2.5. Path change

### 2.2.6. Shutdown

When the transmission is concluded, the source can release the reserved resources by sending a RL\_RESV request. This will cause the intermediate routers to release the allocated resources.

### 2.3. Commands

RESV Sent at the beginning of a session. When it leaves the source it contains the amount of required resources (R\_req) and the minimum admissible amount of resources (R\_min) and the current amount of reserved resources (R\_curr), initialized to R\_req. The request is modified by intermediate routers as follows

- \* If the router cannot allocate R\_curr resources, it writes in R\_curr the reserved amount of resources.
- \* The router creates a new entry in its tables for the new stream and append to the RESV request a "reference" to the new entry.

RESV\_ACK Sent by the target to the source when the target receives the RESV requests. Its payload stores the value of R\_curr in the received RESV request and the list of router references.

KPALV Keep-alive request sent by the source to the target. They carry the values of R\_min, R\_req and R\_curr, together with the router references. They are used, beside for keeping the connection "hot" to that the routers release the resources, for updating the resource reservation. They are ignored by the target node, unless the resource reservation has been changed.

PRL\_ACK Sent by the target to the source when a KPALV with changed reservation is received. Its payload stores the value of R\_curr in the received RESV request and the list of router references.

UP\_RESV Sent by the source to increase the amount of reserved resources. It is handled like a RESV request, with the difference that the stream entry in the router tables must be already present.

RL\_RESV Sent by the source at the end of the session. The routers will release the allocated resources and will remove the stream entry from their tables.

## 2.4. Router behaviour

In this section we describe the expected behavior of a REBOOK-capable router, with reference to a specific implementation. It is clear that the implementation details are mostly a private matter of the router and that they can be changed as long as the router behavior "seen" by the rest of the network does not change.

In the reference implementation chosen here the router uses two tables

- o The usual `_routing table_` mapping destinations to router ports
- o A `_stream table_` addressed via the reference that the router writes in the RESV packet. Each entry of the table contains information relative to the specific stream and it is expected to contain at least
  - \* The current values of `R_min`, `R_req` and `R_curr`
  - \* The reference of the next router
  - \* The session expiration time `T_exp`. If no KPALV packets are received before `T_exp`, the session is considered expired and the allocated resources released
  - \* A reference to the routing table pointing to the entry associated with the target address.

### 2.4.1. On reception of a RESV request

When the router receives a RESV request

- o It creates a new entry in the stream table for the new stream. Let REF be the reference to the new entry. It look-ups the routing information required for the stream and set the routing table reference in the stream table entry accordingly.
- o It checks the amount of resources that it can allocate to the stream and update the stream table accordingly. If the allocated resources are less the `R_curr` value in the RESV request, update the RESV value to the actual amount of allocated resources.
- o It appends to the RESV request the value of REF and forward the updated request to the next hop.

#### 2.4.2. On reception of a data packet with the reference field

As explained in Section 2.2.3, the source can optionally include in the data packet the reference belonging to the first router. In this case, when the router receives a packet that is marked as a REBOOK packet

- o It checks for the packet validity. This is done both for security and for detecting events like route changes and faults. The check can be done in constant time by checking the REBOOK reference in the packet, see Section 2.4.6 for details. If the check is positive, continue, otherwise the REBOOK reference is invalidated and the packet is handled as a normal one.
- o By using the router reference field embedded in the packet, it accesses the entry in the stream table.
- o It copies the router reference stored in the table in the corresponding data packet field. Note that in this way the next REBOOK-capable router will find in the RRef field the reference value that it wrote in the RESV packet during the setup phase.
- o It finds the corresponding entry in the routing table and forward the packet toward the right output port.
- o Finally, note that the access to the table at constant cost allows to implement new features, e.g., accounting and security controls.

Note that the operations above require a constant time, independent on the number of entries in the stream table.

#### 2.4.3. On reception of a KPALV request

When the router receives a KPALV

- o It checks for the packet validity. This is done both for security and for detecting events like route changes and faults. The check can be done in constant time by checking the REBOOK reference in the packet, see Section 2.4.6 for details. If the check is positive, continue, otherwise discard the packet (GIUSTO?)
- o It finds the corresponding entries in the stream table.
- o It updates the expiring time  $T_{exp}$  of the session
- o If the value of  $R_{curr}$  in the KPALV request is lower than the corresponding value stored in the table, it updates the table (and release the corresponding resources too)

- o If the router lowered the resources assigned to the stream because of some resource re-assignment, it update the R\_curr value in the KPALV request
- o The (possibly updated) KPALV request is forwarded to the next hop

#### 2.4.4. On reception of a UP\_RESV request

The behavior is similar to the case of reception of a RESV request, with the difference that the stream must already have an entry in the stream table

#### 2.4.5. On reception of a RL\_RESV request

When the router receives a RL\_RESV

- o It checks its validity (see Section 2.4.6). If the check is positive, continue, otherwise discard the packet (GIUSTO?)
- o It deletes the entry associated to the stream from the stream table and releases the associated resources.

#### 2.4.6. Checking packet validity

For safety reasons, a router must check if the received REBOOK packet is a valid one. The check can be done by exploiting some redundancies in the REBOOK structure

- o When a router receives for the first time a RESV request, it opens a new entry in the stream table and initialize it with the data stored in the request, including the FlowID, that is, the (source address, target address) pair, where each address is, of course, a (host address, port) pair. When a data packet or a REBOOK request is received, the router can check that the FlowID of the received request matches the FlowID stored in the stream table.
- o If the router use only B < 32 bits of the 32 bits of the reference, it can use the remaining 32-B bits for the validity check. A possible approach is to fill the unused 32-B bits with parity bits and encrypting the result. The check for reference validity is immediate and it is very difficult to forge a valid reference. Note that since the reference value is an opaque 32-bit string for the other nodes, this type of checks can be implemented privately by the router without problems of interoperability.
- o Another low-cost validity check supposes that the free entries in the stream table are organized in a linked list. In this case the router initializes the empty stream table by joining its entries

in random order, making unpredictable the order the entries will be used by the router. Moreover, in order to avoid a fast reuse of the same entry, released entries can be inserted at the end of the free list, while allocated entries can be extracted from the head.

In order to check the validity of the REBOOK packet, the router can verify that the referenced entry is really used.

### 3. Using the IP Router Alert field

A possible solution to embed REBOOK information inside data packet is to exploit the IP Router Alert field. This will require the allocation of a suitable code-point in the IANA maintained registry. A REBOOK value in the IP Router Alert field will signal to the router that the 32-bit reference value has been stored between the next level protocol header and the data payload.

### 4. Session Timeout

As already explained, because of the soft state nature of REBOOK, sessions that are not refreshed by keep-alive messages are considered expired and their resources released. An important issue is the choice of the timeout interval: if it is too short, the source will send many keep-alive packets, increasing the overhead; if it is too long, the resources associated with a crashed session (i.e., a session where the source crashed) can remain unavailable for a long time (until the session expires), reducing the efficiency of the router. Note that it is important that the source knows the session timeout, in order to choose the keep-alive frequency.

Some possible solutions about the choice of the session timeout are the following

**Fixed** The timeout value could be fixed by the protocol. This is maybe the simplest solution and it has the advantage of a low overhead, but maybe is too rigid.

**RTT based** The timeout could be fixed to a multiple of the round trip time (RTT) between the source and the target. The source would send keep-alive packet with a frequency that depends on the RTT and the expiration time should be long enough to keep in account the probability that a keep-alive packet gets lost. This overhead of this solution is low. It is important that the difference between the RTT estimated by the router and the RTT estimated by the source is small.



Timeout as a resource With this approach the timeout is considered a resource that can be negotiated by using the REBOOK mechanism (see Section 5). Therefore, the source will send its timeout request in the RESV packet and the routers will change the timeout value in the RESV request according to their needs. Moreover, timeout can be changed dynamically during the session by the usual REBOOK procedure. The drawback of this approach is that it introduces overhead in the RESV and KPALV packets. Note that with this approach, the session timeout is the smallest among the timeouts chosen by the routers.

## 5. What is a resource?

In this document we talked generically about "resources" to be reserved, without describing in details what a resource is. Typically one can expect bandwidth to be a negotiable resource, but other types of resources are possible. For example, the source could want to improve the latency by assigning to the stream a higher priority, so that the router will store the packets in a high-priority queue. Another example of possible resource is the session timeout (see Section 4). Depending on the application using REBOOK, it could prove useful to keep the set of resources extensible, maybe by using a Type-Length-Value format, so that routers that do not recognize some resources can skip them.

## 6. Security consideration

To be written

## 7. IANA considerations

To be written

## 8. References

### 8.1. References

- [RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.
- [RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.

## 8.2. Informative References

[rebook] Montessoro, P. and D. Daniele, "REBOOK: A Deterministic, Robust and Scalable Resource Booking Algorithm", J Netw Syst Manage DOI 10.1007/s10922-010-9167-8, may 2010.

### Authors' Addresses

Pierluca Montessoro  
University of Udine  
Via delle Scienze 208  
Udine 33100  
Italy

Phone: +39-0432-55-8286  
EMail: montessoro@uniud.it

Riccardo Bernardini  
University of Udine  
Via delle Scienze 208  
Udine 33100  
Italy

Phone: +39-0432-55-8271  
EMail: riccardo.bernardini@uniud.it

