Internet Area WG
Internet-Draft
Intended status: Informational
Expires: June 7, 2019

L. Muscariello
G. Carofiglio
J. Auge
M. Papalini
Cisco Systems Inc.
December 04, 2018

Hybrid Information-Centric Networking
draft-muscariello-intarea-hicn-01

Abstract

This documents describes the hybrid information-centric networking (hICN) architecture for IPv6. The specifications describe a way to implement information-networking functionalities into IPv6. The objective is to use IPv6 without creating overlays with a new packet format as an additional encapsulation. The intent of the present design is to introduce some IPv6 routers in the network with additional packet processing operations to implement ICN functions. Moreover, the current design is tightly integrated into IPv6 to allow easy interconnection to IPv6 networks with the additional design objective to exploit existing IPv6 protocols as much as possible as they are, or extend them where needed.

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 https://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 June 7, 2019.

Copyright Notice

Copyright (c) 2018 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 (https://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.

Table of Contents

1. Introduction

The objective of this document is to describe hybrid ICN, a network protocol that integrates ICN in IPv6, at a minimum cost in terms of required modifications in end-points and routers and in a way to guarantee transparent interconnection with IP without using overlays.

The ICN reference design used in this document is CCNx as described in [I-D.irtf-icnrg-ccnxsemantics] and [I-D.irtf-icnrg-ccnxmessages]. IPv6 is used as described in [RFC8200].

There are some basic design principles behind the hICN architecture that are implemented by the design reported below that can be summarized as follows:

- (i) the network can transport many different kinds of applications as IPv6, i.e. hICN can serve content-distribution or real-time applications, to cite examples with very different requirements. hICN is not a content-distribution network;

- (ii) it provides connection-less and location independent communications by identifying data with unique global names, instead of naming network interfaces (locator) or end-hosts (end-host identifiers) as in LISP [RFC6830].

- (iii) data is retrieved by an end-point by issuing requests and a node accepts a data packet from an ingress interface if and only if at least one matching request packet is stored in the local cache of the node, otherwise the data packet is dropped;

- (iv) basic security services are provided by the architecture: authenticity of the data producer and data integrity. A cryptographic signature over a security envelop is computed by the producer (using its own private key) and must be verified by the consumer (using the producer's public key). The security envelop can be as small as a single data packet or cover groups of packets using the technique of the transport manifest [MAN].

2. Architecture

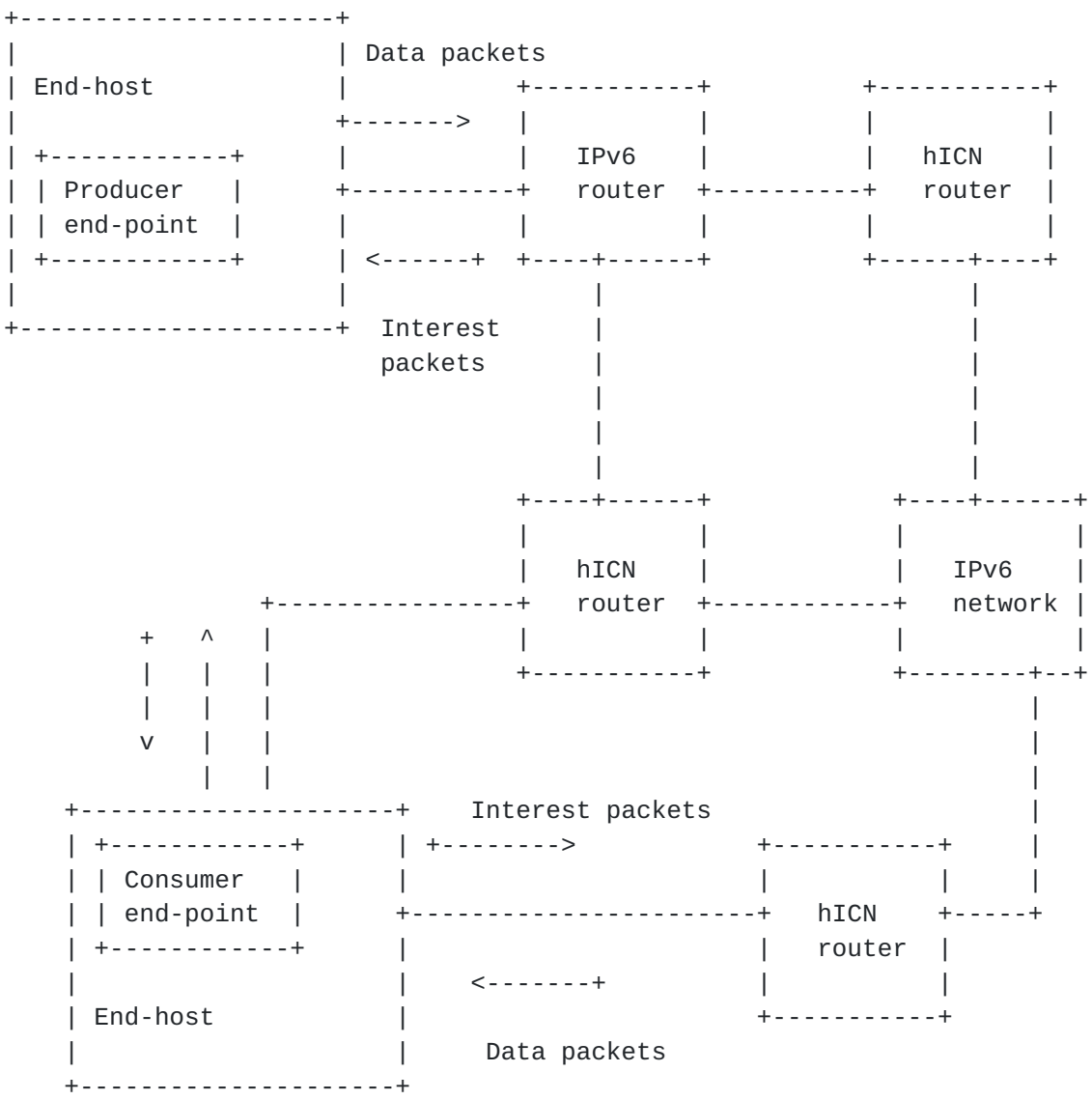

Figure 1: General overview of an hICN end-to-end communication.

The communication model described in this document covers the transport and the network layer.

The network layer includes the forwarding plane only and does not consider the routing plane. hICN network layer is about using the IPv6 FIB to determine a next hop router to forward requests or using a local packet cache to determine if an incoming request can be satisfied locally. The hICN forwarding plane takes care of forwarding replies by using information stored in cached requests. The packet pipeline of an hICN node always includes a lookup in a packet cache for both requests and replies. The packet cache is a mandatory component that is added to the usual IPv6 packet processing pipeline. Requests and replies carry an immutable data name end-to-

end, in packet header fields as described in the following sections. Moreover, requests and replies carry locators as mutable packet header fields. A locator, i.e. an interface identifier, is changed every time a packet is sent to another hICN node. A detailed description of how locators are modified along the path between end-points is reported in the following sections.

It is assumed that existing routing protocols, working for IPv6, should be reused as much as possible as they are. Improvements to existing routing protocols are out of scope and might be developed in other documents to better exploit features made available by the hICN forwarding plane. For instance, hICN forwarding plane can take advantage of the ability of a routing protocol to provide multiple routes for a given destination or more generally compute routes for destinations that are multi-instantiated [MIR]. This topic is important but out of scope for this document.

The hICN network architecture can run on top of any link-layer that supports IPv6. hICN data names are globally routable names which are visible to the transport layer end-points. Conversely, the transport layer has no visibility of addresses of network interfaces. The network layer forwards two kind of protocol data units: the request and the reply, called interest and data packets.

The hICN network layer offers a communication service to the transport layer in the end-points by means of a local unidirectional channel that we call local or application face. This channel is used by the transport layer to send requests and receive replies or to send replies upon receptions of requests.

A transport end-point is always bound to a unidirectional channel that is used to either send data or receive data. The former end-point is called data producer while the latter data consumer. The producer end-point produces data under a location independent name, which is an IPv6 prefix. A consumer end-point fetches data by using the non ambiguous name as provided by the producer. The producer end-point is responsible for managing the usage of the prefix in terms of provisioning, association to applications and its revocation.

The transport end-point offers two kinds of services to applications: a producer and a consumer service. The service is instantiated in the application by opening communication sockets with an API to perform basic transport service operations: allocation, initialization, configuration, data transmission and reception.

The producer and consumer sockets can implement different types of services such as stream or datagram, reliable or unreliable. In all

cases all transport services are connection-less, meaning that a producer transport service produces named data in a socket memory that is accessible by any valid request coming from one or multiple consumers. The consumer, on the other hand, retrieve named data using location independent names which are not tied to any interface identifier (also called locator). This transport model allows to implement reliable consumer mobility without any special mobility management protocol. hICN supports communication of multi-homed end-hosts without any special treatment in the transport layer. The hICN network layer can also implement robust usage of multi-path forwarding in IPv6 networks as balanced request/reply flows self-stabilize network congestion see [CCN],[NDN], [RAQ] .

A data packet is an IPv6 packet with a transport layer header carrying data from an application that produces data. An interest packet is an IPv6 packet with a transport layer header and is used to unambiguously fetch a data packet from a producer end point.

2.1. End-points

In hICN we introduce two new kinds of endpoints: the producer and the consumer. We identify two kind of communication sockets each with a specific API: the producer and consumer sockets. These socket types are designed to exchange data in a multi-point to multi-point manner. In (h)ICN we have the same concept that is applied to a network where memories are distributed across the communication path. The first memory in the path is the production buffer of the producer end-point that forges Data Packets and copies them into a shared memory isolated into a namespace. Consumer sockets can retrieve data from such memory by using the (h)ICN network layer. The model just described is an inter-process communication example (IPC) that requires data to cross a communication network by using a transport protocol.

The way consumers and producers synchronize depends on application requirements and the transport layer exposes a variety of services: stream/datagram, reliable/unreliable, with or without latency budgets etc. Independently of the specific requirements of the applications, producer sockets always perform data segmentation from the upper layer into Data Packets, as well as compute digital signatures on the packet security envelop. This envelop can also be computed across a group of packets, by including a cryptographic hash of each packet into the transport manifest, and eventually signing only such manifest.

The consumer socket, on the other end, always performs reassembly of Data Packets, hash integrity verification and signature verification. This is common to all architectures in (h)ICN. The usual assumption

is that the producer socket uses an authentic-able identity while using namespaces that it has been assigned. The end-point must be able to manage the mapping of her identity and the allocated namespace by issuing digital certificates about the mapping. The consumer end-point must retrieve the associated certificate to perform the basic operations. It is out of scope for this document how to design and implement a scalable system to perform such certificate operations.

A detailed description of transport end-points is out of scope for this document. A detailed description of transport end-points is out of scope for this document but more details can be found in [TRA].

2.2. Naming

In hICN, two name components are defined: the name prefix and the name suffix. The name prefix is used to identify an application object, a service or in general an application level source of data in the network. This is incarnated by a listening socket that binds to the name prefix. The name suffix is used to index segmented data within the scope of the name prefix used by the application.

For instance an RTP [RFC3550] source with a given SSRC can be mapped into a name prefix. Single RTP sequence numbers can be mapped into name suffixes. For example an HTTP server can listen to a name prefix to serve HTTP requests. An HTTP reply with large payload with require the transport layer to segment the application data unit according to an MTU. Name suffixes are used to index each segment in the socket stream.

More details about how to use hICN to transport HTTP or RTP will be given in a different document.

2.2.1. Name prefix

The format of an hICN name prefix is the following:

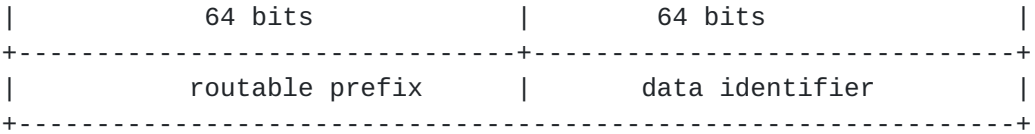

Figure 2: hICN IPv6 name prefix.

It is composed of a routable IPv6 /64 prefix as per [RFC3587] which SHOULD be globally routable. The data identifier is encoded in 64 bits. An application can use several identifiers if needed.

From the description given above, the name prefix is a location independent name encoded in an IPv6 address.

2.2.2. Name Suffix

The name suffix is used by the transport layer protocol to index segments. The segment MUST be indexed in the end-points and in the network with the same suffix. This implies that there is one transport segment per IP packet and that IP fragmentation is not allowed. Extension to allow secure fragmentation are possible, such as [FRA] but they are out of scope for this document. It is up to the producer end-point to determine how to perform segmentation depending on the use case. An MTU path discovery protocol for hICN is out of scope of this document and additional work is required to extend existing protocols or design new ones.

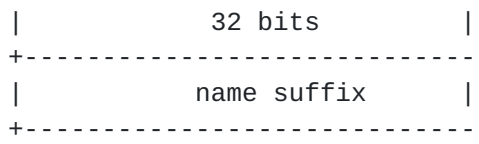

Figure 3: hICN name suffix.

2.3. Packet Format

Two protocol data units are defined below: the interest (request) and the data (reply).

They are composed of a network and transport header. The transport header is the same for both packet types while the network header is slightly different.

2.3.1. Interest Packet

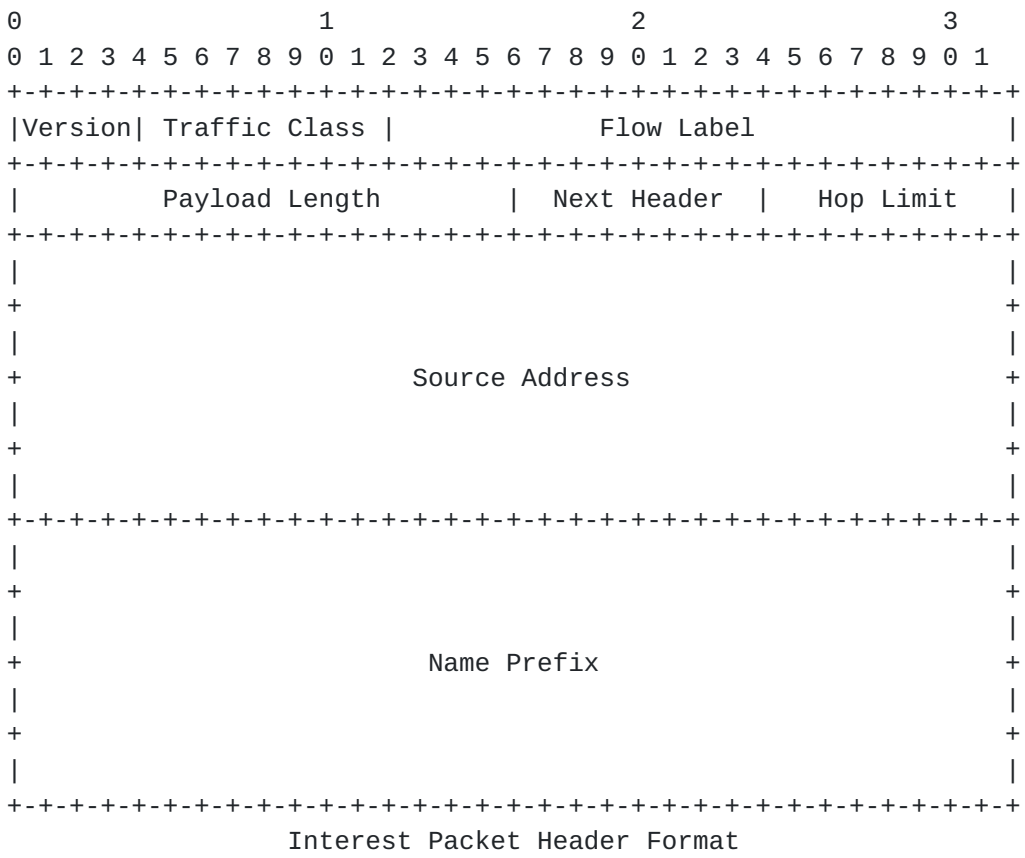

Figure 4: IPv6 interest packet L3 header.

Source Address: 128-bit address of the originator of the packet (possibly not the end-host but a previous hICN node).

Name Prefix: 128-bit name prefix of the intended service.

2.3.2. Data Packet

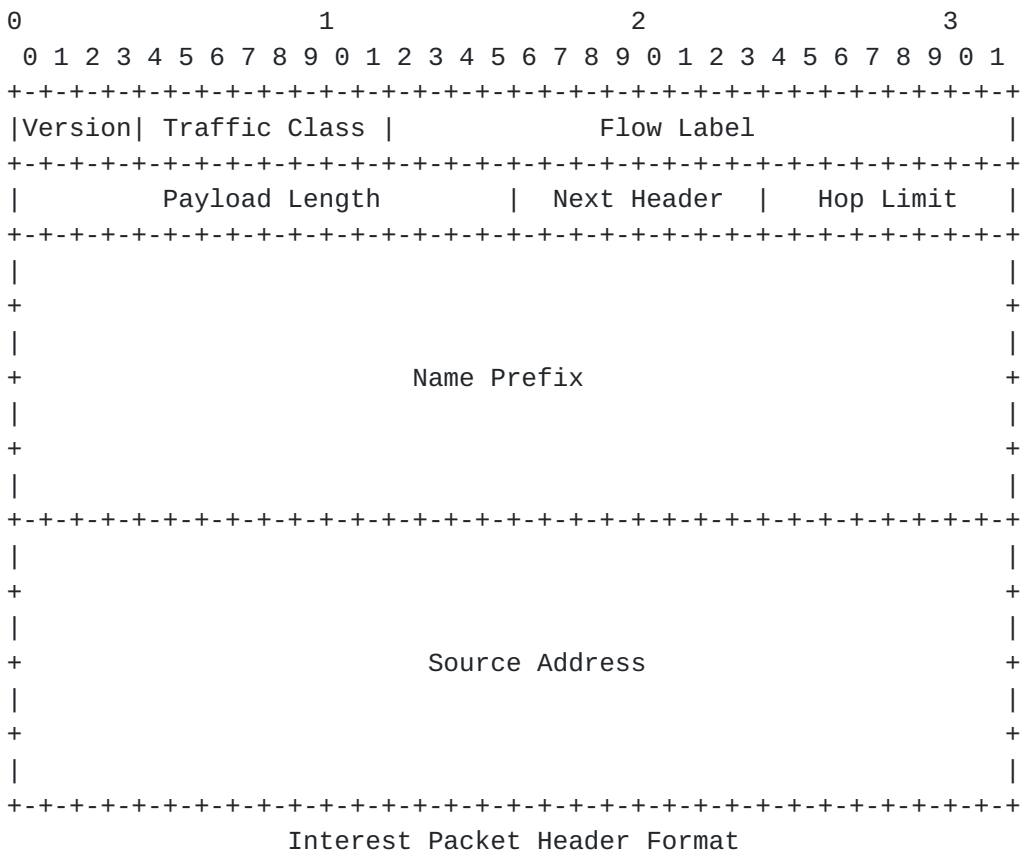

Figure 5: IPv6 data packet L3 header.

Name Prefix: 128-bit name prefix of the intended service.

Source Address: 128-bit address of the destination of the packet (possibly not the end-host but the next hICN node).

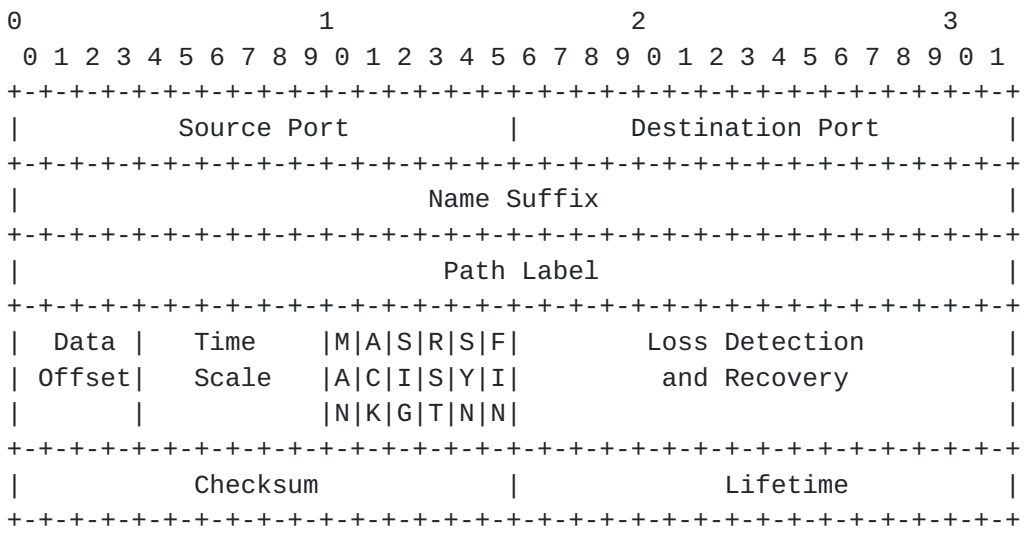

Figure 6: Transport header for data and interest packets.

Name Suffix: 32-bit name-suffix of the packet (possibly not the end-host but a previous hICN node).

Path Label: 32-bit label used to carry an encoding of the path between the consumer and data responder, be it an intermediate node or the producer end-point.

Time Scale: 6-bit natural number in the range 1-64 used as a scaling factor for time calculations. If not null it is used to scale lifetime.

Manifest: flag to indicate the packet carries a transport manifest in the payload.

Signature: flag to indicate the packet carries an authentication header with a signature. Interest packet do not carry signatures.

Loss Detection and Recovery: 16-bit natural number used to implement data sequencing on per adjacency basis to detect an recover losses using the mechanism WLDR described in {{WLD}}.

Lifetime: 16-bit unsigned integer to carry the packet lifetime in milliseconds.

Checksum: Updated using RFC 1624.

The following sections describe the components of an hICN node and the packet processing operations.

2.4. Packet cache

The packet cache is a router local memory used to temporarily store requests and reply. The simplest incarnation of the packet cache MUST index packets by full name, i.e. the concatenation of the name prefix and suffix. Insertion and deletion of packets in the cache is described below.

2.5. Forwarding

The forwarding path in hICN is composed of two components: the interest and data path. Requests and replies are processed at the hICN node in a different way. Both forwarding paths require a packet cache to be incorporated into the router. The cache is used to temporarily store requests and replies for a relatively short amount of time.

By caching a request in an hICN node, the reply can be transmitted back to the right nodes as the source address field in the interest contains the interface identifier of the hICN node having transmitted the request. Replies are optionally cached if needed.

This means that the interest forwarding path is based on lookups in the IP FIB just like any other IP packet, with the additional processing due to a cache lookup to check if the actual reply is already present in the local cache for expedited reply.

On the other hand, data packet forwarding is similar to label swapping [RFC3031], being the packet name identifier (prefix plus suffix) the forwarding label. The next hop for the reply in transit is indeed selected by using information in a cached matching request.

The name prefix in the packet header is never modified along the path for both requests and replies, while the locator, i.e. the interface identifier written in the source or destination address field, for interest or data packets respectively, is modified at the egress of the router as reported below.

2.5.1. Interest Path

At the router ingress the incoming interest packet I is parsed to obtain the name prefix and the name suffix. An exact match look up is made in the packet cache using the full packet name as key. Based on the outcome of the lookup the following options are possible:

1. at least one match is found.

1.1. If one match is a data packet D, other matches are ignored, and D is prepared for transmission by setting D's destination address with I's source address. D is passed to the egress to further processing before transmission. For instance the next-hop MAY be selected by using the router IPv6 FIB (longest prefix match). The IPv6 FIB lookup MIGHT be saved in case the next-hop can be derived directly from information previously derived by processing the incoming interest packet I. I is eventually dropped.

1.2. There is one or multiple matches and all are interest packets.

* One matching interest has the same source address and I is classified as duplicate and further processed as duplicate.

* Matching interest packets have different source addresses and I is classified as filtered and stored in the cache.

2. a match is not found and I passed to the egress for further processing to determine the next-hop by using the router IP FIB.

Notice that the destination address field in the interest packet is polymorphic as it has two different types based on the data structured it is looked-up against. It has the type of a location independent name while used to find a match in the packet cache and it has an address prefix type to find the next-hop in the IPv6 FIB. Polymorphism is transparent for the forwarding plane while it has several implications in the control plane.

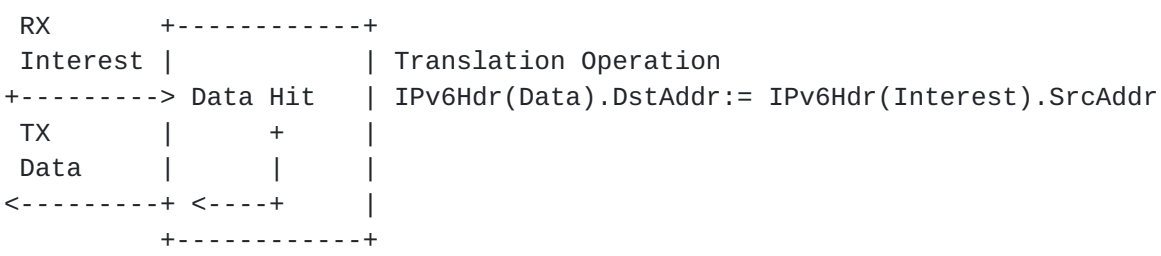

Figure 7: The interest packet hits a matching data packet in the packet cache.

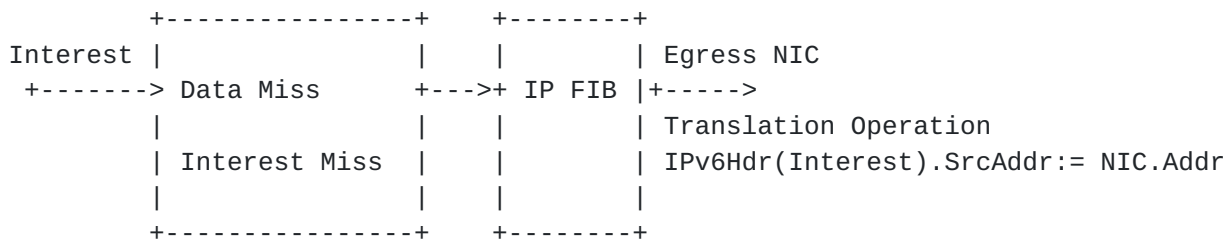

Figure 8: The interest packet finds no match in the packet cache and is processed to find a next-hop.

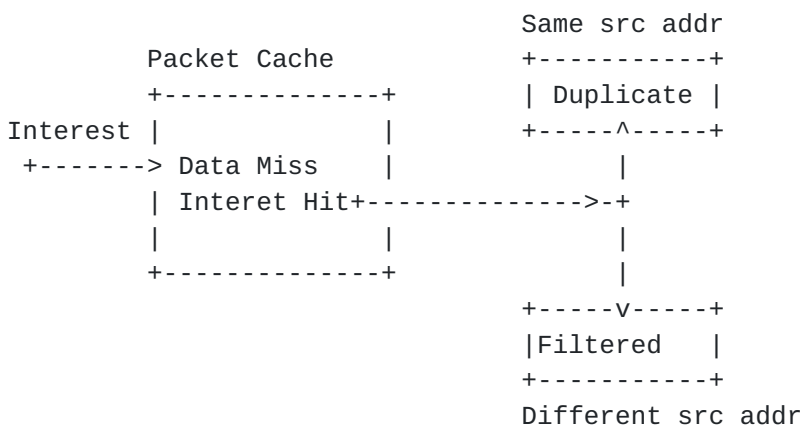

Figure 9: The interest packet hits an interest packet in the packet cache.

2.5.2. Data Path

At the router ingress the incoming data packet D is parsed to obtain the name prefix and the name suffix. An exact match look up is made in the packet cache using the full packet name as key. Based on the outcome of the lookup the following options are possible:

1. one or multiple matching interest packets are found 1.1. The data packet D is cloned to have as many copies as the number of matching interests including D. The destination address field of each copy of D is set with the source address field of each interest packet. All copies are passed to the egress to further processing before transmission in order to find each data packet's next-hop.

2. No matching is an interest packet and the D is dropped.

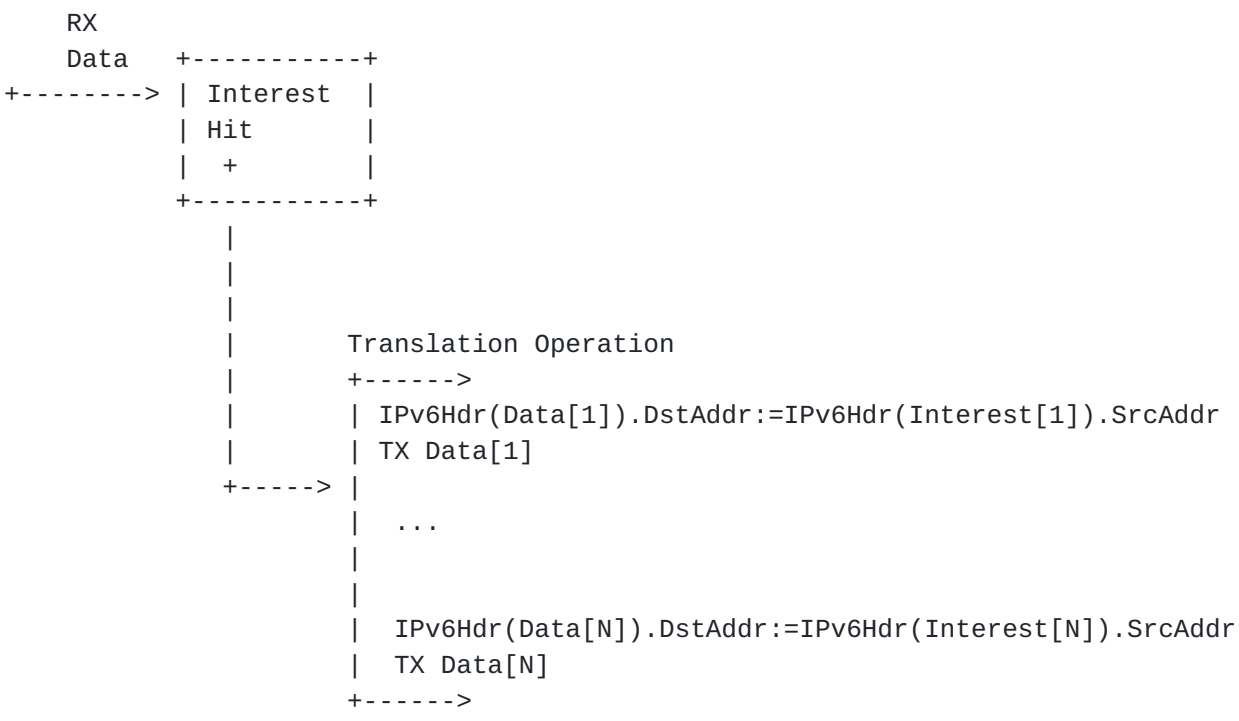

Figure 10: The data packet hits an interest packet in the packet cache.

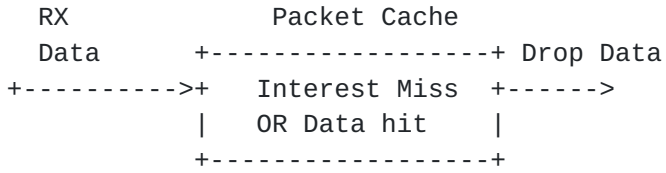

Figure 11: The data packet is drop in case no interest match is found in the packet cache.

3. Security

hICN inherits ICN data-centric security model: integrity, data-origin authenticity and confidentiality are tied to the content rather than to the channel.
Integrity and data-origin authenticity are provided through a digital signature computed by the producer and included in each data packet. Integrity and data-origin authenticity are provided in two ways using two approaches: the first one based on IP Authenticated Header [RFC4302] and the second one based on transport manifests. Notice that the IP AH is not used as an IPv6 extension header as it is appended after the transport header. However the choice of the IP AH

has been made in order to exploit existing protocol implementations in the end-points.

When using IP AH, the signature is computed over

- (i)IP or extension header fields either immutable in transit or that are predictable in value upon arrival at the consumer,
- (ii) the AH header with the signature field set to zero. We recall that in hICN the destination header field is not immutable nor predictable and must be set to zero for the signature computation. We also point out the AH in placed after the TCP header in order to prevent any kind of filtering from network devices such as middleboxes.

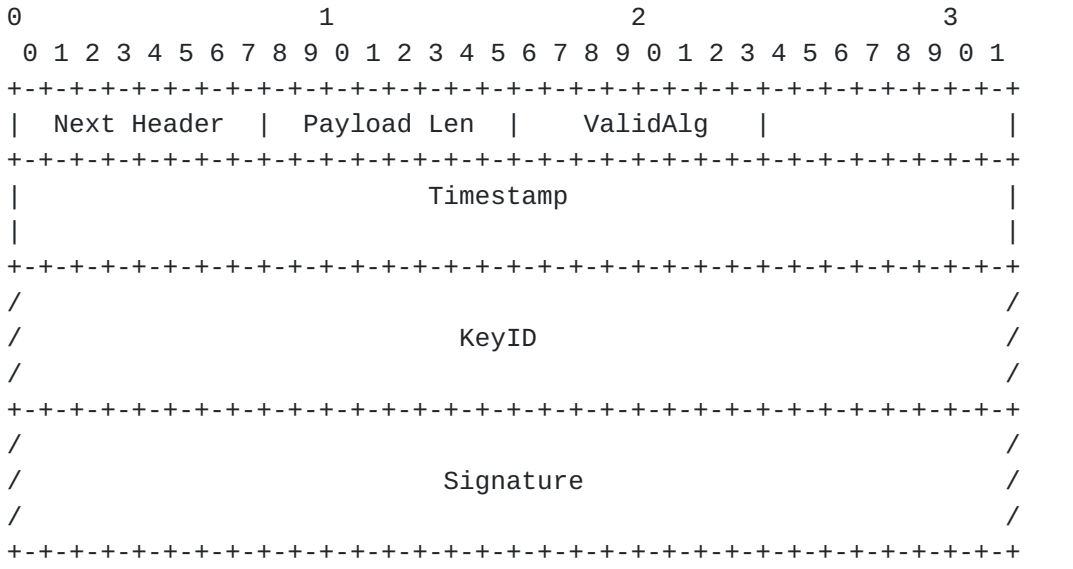

Figure 12: The IP authentication header appended after the transport header to carry packet signatures.

ValidAlg: 8-bit index to indicate which validation algorithm must be used to verify the signature.

Timestamp: 64-bit time stamp that indicates the validity of the signature.

KeyID: 256-bit key identifier.

Signature: Variable length field that carries the cryptographic signature of the security envelope.
It is 128 bytes for RSA-1024, 256 bytes for RSA-2048, 56 bytes for EDCSA 192, 72 bytes for ECDSA 256.

The transport manifest is a L4 entity computed at the producer which contains the list of names of a group of data packets to convey to the consumer. hICN cryptographic hashes of data packets are then computed instead of signatures. The hashes are computed on immutable fields as explained above when using the IP AH. Moreover, the manifest must be signed to guarantee a level of security equivalent to packet-wise signatures.

hICN is oblivious of the trust model adopted by consumers and works with any of the existing proposals.

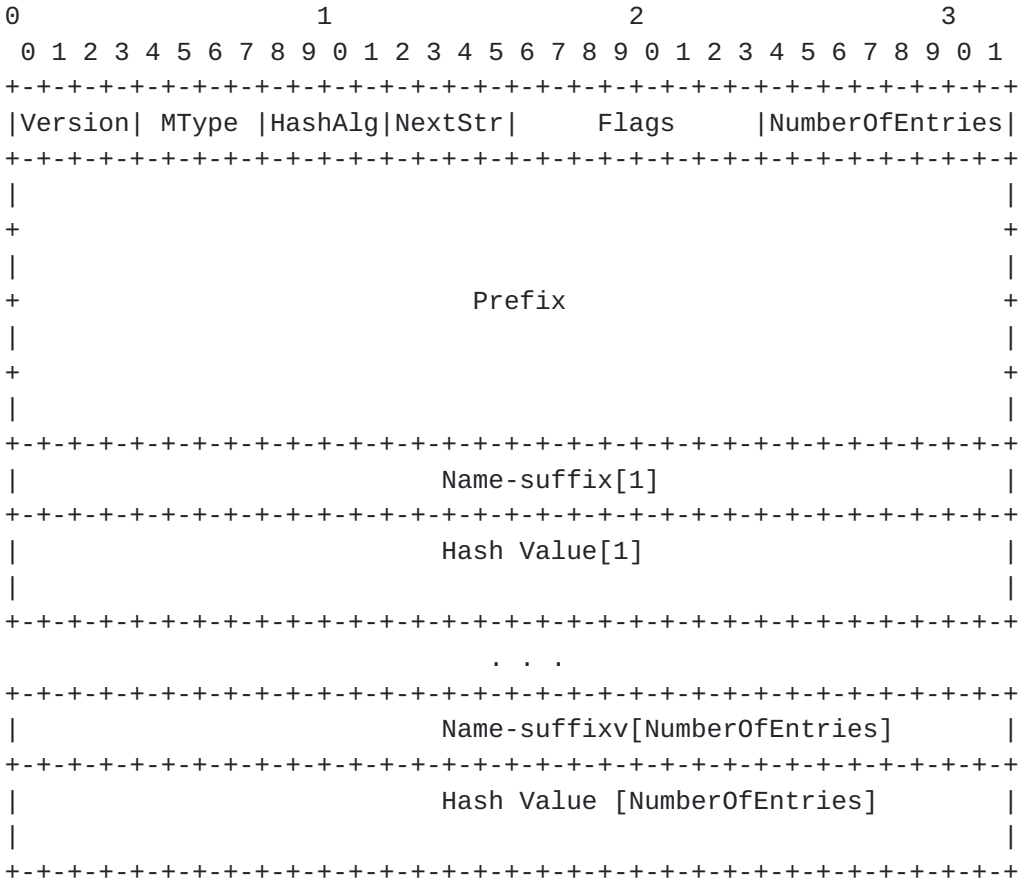

Figure 13: The transport manifest, generated by the producer end-point for the consumer end-point, contains names, integrity hashes and is signed with the producer end-point private key

Version:	8-bit index to indicate which validation algorithm must be used to verify the signature.
MType:	64-bit time stamp that indicates the validity of the signature.
HashAlg:	256-bit key identifier.
NextStr:	Encode an operator use to predict the name-suffix sequence
Flags:	Flags.
NumberOfEntries:	8-bit field that encodes the number of packets indexed in the manifest.
Name-prefix:	128-bit field carrying the name-prefix common to all packets indexed in the manifest.
Name-suffix:	32-bit field carrying the name-suffix.
Hash-value:	256-bit field carrying the SHA-256 hash of the packet security envelop.

4. The End-host model and End-to-End considerations

In hICN the end-host model is very similar to a regular IPv6 end-host with some extensions. An end-host is capable of opening consumer and producer transport end-points, one to receive data and one to send data under a given name prefix. The end-host continues to identify interfaces using IPv6 addresses (locators or routing locators, RLOCs, using LISP terminology), just like any IPv6 router. In addition to that, transport end-points bind to location-independent names, similar to LISP end-point identifiers (EIDs). However, instead of using name prefixes to identify end-hosts only, in hICN a name prefix is used to identifity a data source.

There is an analogy between IPv6 multicast and the hICN data forwarding path for one-to-many communications, as the IPv6 multicast group address identifies data that group members receive from a single sender. Notice that in hICN a data packet transmission stores the identifiers in the source address field while in IPv6 multicast it is stored in the destination address field.

Theres is also an analogy between IPv6 anycast and the hICN interest forwarding path, where multiple interfaces make use of the same IPv6 (anycast) address. Multiple instances of the same applications can

then run at different end-points to eventually reply to the same request.

An hICN network node behaves as an end-host consumer end-point for the upstream producer end-point as all replies are forced to flow back to the same hICN that transmitted the requests. An hICN network node may be able to reply to a request on behalf of a end-point producer, in that case that hICN node behaves as an end-host for the consumer end-point.

5. IANA Considerations

There are no IANA considerations in this specification.

6. Acknowledgements

The authors would like to thank David Ward, David Oran, Paul Polakos, Mark Townsley, Mauro Sardara and Alberto Compagno for suggestions on how to improve the architecture and the current document.

7. References

7.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1081] Rose, M., "Post Office Protocol: Version 3", RFC 1081, DOI 10.17487/RFC1081, November 1988, <https://www.rfc-editor.org/info/rfc1081>.

[RFC1624] Rijsinghani, A., Ed., "Computation of the Internet Checksum via Incremental Update", RFC 1624, DOI 10.17487/RFC1624, May 1994, <https://www.rfc-editor.org/info/rfc1624>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <https://www.rfc-editor.org/info/rfc3587>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

7.2. Informative References

[CCN] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking named content", Proceedings of the 5th international conference on Emerging networking experiments and technologies - CoNEXT '09, DOI 10.1145/1658939.1658941, 2009.

[FRA] Mosko, M. and C. Wood, "Secure Fragmentation for Content Centric Networking", 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, DOI 10.1109/mass.2015.51, October 2015.

[I-D.irtf-icnrg-ccnxmessages] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-08 (work in progress), July 2018.

[I-D.irtf-icnrg-ccnxsemantics] Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", draft-irtf-icnrg-ccnxsemantics-09 (work in progress), June 2018.

[I-D.irtf-icnrg-mapme]
Auge, J., Carofiglio, G., Muscariello, L., and M. Papalini, "MAP-Me : Managing Anchorless Mobility in Content Centric Networking", draft-irtf-icnrg-mapme-02 (work in progress), October 2018.

[I-D.irtf-icnrg-terminology]
Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology", draft-irtf-icnrg-terminology-01 (work in progress), October 2018.

[MAN] Baugher, M., Davie, B., Narayanan, A., and D. Oran, "Self-verifying names for read-only named data", 2012 Proceedings IEEE INFOCOM Workshops, DOI 10.1109/infcomw.2012.6193505, March 2012.

[MIR] Garcia-Luna-Aceves, J., Martinez-Castillo, J., and R. Menchaca-Mendez, "Routing to Multi-Instantiated Destinations: Principles, Practice, and Applications", IEEE Transactions on Mobile Computing Vol. 17, pp. 1696-1709, DOI 10.1109/tmc.2017.2734658, July 2018.

[NDN] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., and B. Zhang, "Named data networking", ACM SIGCOMM Computer Communication Review Vol. 44, pp. 66-73, DOI 10.1145/2656877.2656887, July 2014.

[RAQ] Carofiglio, G., Gallo, M., Muscariello, L., Papalini, M., and , "Optimal multipath congestion control and request forwarding in Information-Centric Networks", 2013 21st IEEE International Conference on Network Protocols (ICNP), DOI 10.1109/icnp.2013.6733576, October 2013.

[TRA] "M. Sardara, L. Muscariello and A. Compagno, A Transport Layer and Socket API for (h)ICN: Design, Implementation and Performance Analysis, In Proc. of ACM SIGCOMM ICN", 2018.

[WLD] Carofiglio, G., Muscariello, L., Papalini, M., Rozhnova, N., and X. Zeng, "Leveraging ICN In-network Control for Loss Detection and Recovery in Wireless Mobile networks", Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16, DOI 10.1145/2984356.2984361, 2016.

Authors' Addresses

Luca Muscariello
Cisco Systems Inc.

Email: lumuscar@cisco.com

Giovanna Carofiglio
Cisco Systems Inc.

Email: gcarofig@cisco.com

Jordan Auge
Cisco Systems Inc.

Email: augjorda@cisco.com

Michele Papalini
Cisco Systems Inc.

Email: mpapal@cisco.com