ICNRG
Internet-Draft
Intended status: Experimental
Expires: December 31, 2015

M. Mosko
I. Solis
PARC, Inc.

June 29, 2015

CCNx Semantics
draft-irtf-icnrg-ccnxsemantics-00

Abstract

This document describes the core concepts of the CCNx architecture and presents a minimum network protocol based on two messages: Interest and Content Object. It specifies the set of mandatory and optional fields within those messages and describes their behavior and interpretation. This architecture and protocol specification is independent of a specific wire encoding.

The protocol also uses a Control message called an InterestReturn, whereby one system can return an Interest message to the previous hop due to an error condition. It indicates to the previous hop that the current system will not respond to the Interest.

Table of Contents

1. Introduction

This document describes the principles of the CCNx architecture. It describes the network protocol based on two message types: Interests and Content Objects. The description is not dependent on a specific wire format or particular encodings.

CCNx uses subjective names to identify bytes of payload. The Name combines a routable prefix with an arbitrary suffix assigned by the publisher to a piece of content. The result is a "named payload". This is different from other systems that use only self-certifying names, where the payload name is intrinsically derivable from the payload or its realization in a network object (e.g. a SHA-256 hash of the payload or network object).

The key concept of CCNx is that a subjective name is bound to a fixed payload via cryptographic operations. This implies that the fact that a given publisher bound a certain subjective name to a certain payload can be verified via cryptographic means. For example, a publisher could use a cryptographic hash over the name and payload, sign the hash, and deliver the tuple {Name, Payload, Validation}. Additional information would be included as needed by different validation mechanisms are used. A typical named payload is thus {Name, Payload, ValidationAlgorithm}.

CCNx specifies a network protocol around Interests (request messages) and Content Objects (response messages) to move named payloads. An Interest includes the Name, the desired payload, and two optional restrictions to limit responses to a specific publisher or a specific Content Object. The Content Object response carries a matching Name and the specified payload. Matching a Content Object to an Interest is an exact match on the Name. The CCNx network protocol of Interests and Content Objects imposes a restriction on Names: each Name should be hierarchical and is used to route towards an authoritative source. The CCNx Name looks like a URI absolute path and we use URI terminology to describe CCN Names as made up of name segments. In practice it has a binary encoding, not a text string.

The hierarchy of a CCNx Name is used for routing via the longest matching prefix in a Forwarder. The longest matching prefix is computed name segment by name segment in the hierarchical path name, where each name segment must be exactly equal to match. There is no requirement that the prefix be globally routable. Within a deployment any local routing may be used, even one that only uses a single flat (non-hierarchical) name segment. Some Forwarders may use more advanced matching rules that allow both longest matching prefix and shorter prefixes.

Another central concept of CCNx is that there should be flow balance between Interest messages and Content Object messages. At the network level, an Interest traveling along a single path should elicit no more than one Content Object response. If some node sends the Interest along more than one path, that node should consolidate the responses such that only one Content Object flows upstream from it to the requester.

There are additional optional attributes in an Interest message that can be used to select between multiple Content Objects with matching Names (it is possible that multiple publishers could issue Content Objects with the same Name). An Interest may carry an optional KeyIdRestriction and / or an optional ContentObjectHashRestriction. If either or both of these are present, a Forwarder must ensure that they exactly match the corresponding KeyId and computed ContentObjectHash in the Content Object.

As an Interest travels the forward path following the Forwarding Information Base (FIB), it leaves behind state at each Forwarder. This state is stored in the Pending Interest Table (PIT), which tracks the ingress and egress ports as well as the Name, KeyIdRestriction (if one exists), and ContentObjectHashRestriction (if one exists) of each Interest. When a Content Object arrives at the node, it is exactly matched against that tuple to see if it satisfies any Interests in the PIT. If it does, it is returned along the matching Interest's reverse path. If it does not, the Content Object is dropped.

If multiple Interests with the same tuple {Name, KeyIdRestriction, ContentObjectHashRestriction} arrive at a node before a Content Object matching the first Interest comes back, they are grouped in the same PIT entry and their reverse paths aggregated. Thus, one Content Object might satisfy multiple pending Interests.

In CCNx, higher-layer protocols often become so-called "name-based protocols" because they operate on the CCNx Name. For example, a versioning protocol might append additional name segments to convey state about the version of payload. A content discovery protocol might append certain protocol-specific name segments to a prefix to discover content under that prefix. Many such protocols may exist and apply their own rules to Names. They may be layered with each protocol encapsulating (to the left) a higher layer's Name prefix.

This document also describes a control message called an InterestReturn. A network element may return an Interest message to a previous hop if there is an error processing the Interest. The returned Interest may be further processed at the previous hop or returned towards the Interest origin. When a node returns an

Interest it indicates that the previous hop should not expect a response from that node for the Interest -- i.e. there is no PIT entry left at the returning node for a Content Object to follow.

There are multiple ways to describe larger objects in CCNx. Some options may use the namespace while others may use a structure such as a Manifest. This document does not address these options at this time.

The remainder of this document describes a named payload, and the Interest/Content Object network protocol behavior in detail.

TODO -- we have not adopted the Requirements Language yet.

1.1. Requirements Language

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].

2. Named Payload

CCNx supports several cryptographic means to bind a Name to a payload. Interest and Content Object messages include a section called the ValidationAlgorithm, which specifies the algorithm to use to verify the binding. Several validation algorithms are supported including specific Message Integrity Checks (MICs), Message Authentication Codes (MACs), and Signature types. These are over specific wire format encodings. Additional schemes could be added in the future. Interest and Content Object messages also include a section called the ValidationPayload which contains the validation output.

The KeyId is an optional field in the ValidationAlgorithm. It is an octet string that identifies the key used to sign the Content Object. It uniquely identifies the publisher. It is similar to a Subject Key Identifier from X509 [RFC 3280, Section 4.2.1.2]. It should be derived from the key used to sign, for example SHA-256 hash of the key. It applies to both public/private key systems and to symmetric key systems using HMAC.

A PublicKeyLocator is an optional field in the ValidationAlgorithm. It may be one of (a) the signer's public key, (b) the signer's certificate, or (c) a KeyName that points to the location of the signer's key or certificate.

A Key inside a PublicKeyLocator is a public key corresponding to the signer's private key. Examples would be PEM or DER encodings. The exact encoding is up to the wire format.

A Certificate is an X.509 certificate of the signer's public key. Examples would be PEM or DER encodings of the certificate. The exact encoding is up to the wire format.

A KeyName is a CCNx Name and optional signer's KeyId of that name's publisher. The CCNx Name points to a Key or Certificate. The KeyName signer KeyId is of the signer of the target name's Content Object, not of the target key or certificate.

A SignatureTime is an optional field in the ValidationAlgorithm. It may be used as part of the signature validation check to ensure the signature was generatred around the expected time. In particular, if the public key is conveyed in a certificate with a validity period, the verifying system may enforce that the signature came from a corresponding period. Some verifiers might determine a signature without a SignatureTime to be invalid.

3. Names

A CCNx name is a composition of name segments. Each name segment carries a label identifying the purpose of the name segment, and a value. For example, some name segments are general names and some serve specific purposes, such as carrying version information or the sequencing of many chunks of a large object into smaller, signed Content Objects.

The name segment labels specified in this document are given in the table below. Name Segment is a general name segment, typically occurring in the routable prefix and user-specified content name. Other segment types are for functional name components that imply a specific purpose.

A forwarding table entry may contain name segments of any type. Routing protocol policy and local system policy may limit what goes into forwarding entries, but there is no restriction at the core level. An Interest routing protocol, for example, may only allow binary name segments. A load balancer or compute cluster may route through additional component types, depending on their services.

Name	Description
Name Segment	A generic name segment that includes arbitrary octets.
Interest Payload ID	An octet string that identifies the payload carried in an Interest. As an example, the Payload ID might be a hash of the Interest Payload. This provides a way to differentiate between Interests based on the Payload solely through a Name Segment without having to include all the extra bytes of the payload itself.
Application Components	An application-specific payload in a name segment. An application may apply its own semantics to these components. A good practice is to identify the application in a Name segment prior to the application component segments.

Table 1: CCNx Name Segment Types

At the lowest level, a Forwarder does not need to understand the semantics of name segments; it need only identify name segment boundaries and be able to compare two name segments (both label and

value) for equality. The Forwarder matches paths segment-by-segment against its forwarding table to determine a next hop.

4. Interests

An Interest is composed of the tuple {Name, Metadata, Payload, Validator}. These fields are defined below. Only the Name is mandatory. Other fields, if missing, should be interpreted as "do not care".

Name is a hierarchical path that identifies the resource. It is matched as described above.

An Interest may also have a Payload which carries state about the Interest but is not used to match a Content Object. If an Interest contains a payload, the Interest name should contain an Interest Payload ID (IPID). The IPID allows a PIT table entry to correctly multiplex Content Objects in response to a specific Interest with a specific payload ID. The IPID could be derived from a hash of the payload or could be a GUID or a nonce. An optional Metadata field defines the IPID field so other systems could verify the IPID, such as when it is derived from a hash of the payload. No system is required to verify the IPID.

An Interest may contain Validation fields including a ValidationAlgorithm section describing the type of validator to use and the ValidationPayload fields containig the output of the validation. Typically this would only be a MIC - a crc, checksum, or digest.

An Interest contains additional fields with information about the query. Two fields - the KeyIdRestriction and ContentObjectHashRestriction - serve as selectors to help identify the specific Content Object that should be returned.

The Interest Lifetime element specifies the maximum number of milliseconds a Forwarder should retain the Interest in its PIT. A lifetime of "0" means the requester does not expect a response from the Interest - it serves as a type of notification. The lifetime is only a guideline for a Forwarder, which may keep an Interest for a shorter or longer time, based on local conditions and system policy. It may change hop to hop if the Interest is delayed for any signifiant amount of time. It is measured in millisecond resolution, so in fast switching it normally would not need to change. This field does not affect the matching of Content Objects.

The Interest HopLimit element is a counter that is decremented with each hop. It limits the distance an Interest may travel. The node originating the Interest may put in any value - up to the maximum - in network byte order. Each node that receives an Interest with a HopLimit decrements the value upon reception. If the value is 0

after the decrement, the Interest cannot be forwarded off the node. The PIT entry should also track the maximum HopLimit forwarded. If an Interest with a longer HopLimit arrives, it should be forwarded even if it is identical to a previously forwarded Interest.

Other implementations may define additional optional headers, for example Nonces for loop detection, headers for Differentiated Services Code Points (DSCP), or Flow Labels.

5. Content Objects

A Content Object is composed of the same tuple as an Interest: {Name, Metadata, Payload, Validator}.

The Name is a mandatory field that identifies the contents.

The Payload of a Content Object holds the upper layer payload. It may be encrypted, based on outside information or a protocol information header.

The optional Metadata PayloadType field identifies the type of Content Object: "DATA" implies an opaque payload; "LINK" is a Content Object with an encoded Link as a payload. "DATA" is the default.

The optional field ExpiryTime is a millisecond timestamp containing the number of milliseconds since the epoch in UTC of when the contents will expire. If it is not present, there is no expiration on the Content Object. The ExpiryTime should be part of the signed information covered by the Validator, if present.

A publisher or upstream node may include a Recommended Cache Time for Content Objects. It is represented as the number of milliseconds since the epoch in UTC as the desired minimum time to keep the content object in cache. This recommendation is a guideline as to the useful lifetime of a Content Object, but may be ignored. The RecommendedCacheTime should not be covered by the Validator, if present, as nodes may change the value based on their caching. The ExpiryTime takes precedence over the RecommendedCacheTime, if both exist.

Other protocols, such as versioning or chunking, could place other kinds of metadata in the Content Object.

6. Link

A Link is the tuple {Name, KeyId, ContentObjectHash}. The information in a Link comprises the fields the fields of an Interest which would retrieve the Link target. A Content Object with PayloadType = "Link" is an object whose payload is a Link. This tuple may be used as a KeyName to identify a specific object with the certificate wrapped key.

7. Validation

7.1. Validation Algorithm

The Validator consists of a ValidationAlgorithm that specifies how to verify the message and a ValidationPayload containing the validation output. The ValidationAlgorithm section defines the type of algorithm to use and includes any necessary additional information. The validation is calculated from the beginning of the CCNx Message through the end of the ValidationAlgorithm section. The ValidationPayload is the actual cryptographic bytes, such as a CRC value or an HMAC value or a signature value.

Some Validators contain a KeyId, identifying the publisher authenticating the Content Object. If an Interest carries a KeyIdRestriction, then that KeyIdRestriction MUST exactly match the Content Object's KeyId.

Validation Algorithms fall into three categories: MICs, MACs, and Signatures. Validators using MIC algorithms do not need to provide any additional information; they may be computed and verified based only on the algoritm (e.g. CRC32C). MAC validators require the use of a KeyId identifying the secret key used by the authenticator. Because MACs are usually used between two parties that have already exchanged secret keys via a key exchange protocol, the KeyId may be any agreed-upon value to identify which key is used. Signature validators use public key cryptography such as RSA, DSA, or Elliptical Curve (EC). The KeyId field in the ValidationAlgorithm identifies the public key used to verify the signature. A signature may optionally include a KeyLocator, as described above, to bundle a Key or Certificate or KeyName. MAC and Signature validators may also include a SignatureTime, as described above.

A PublicKeyLocator KeyName points to a Content Object with an X509 certificate in the payload. In this case, the target KeyId must equal the first object's KeyId. The target KeyLocator must include the public key corresponding to the KeyId. That key must validate the target Signature. The payload is an X.509 certificate whose public key must match the target KeyLocator's key. It must be issued by a trusted authority, preferably specifying the valid namespace of the key in the distinguished name.

8. Interest to Content Object matching

A Content Object satisfies an Interest if and only if (a) the Content Object name exactly matches the Interest name, and (b) the ValidationAlgorithm KeyId of the Content Object exactly equals the Interest KeyIdRestriction, if given, and (c) the computed ContentObjectHash exactly equals the Interest ContentObjectHashRestriction, if given. A Content Object does not carry the ContentObjectHash as an expressed field, it must be calculated in network to match against. It is sufficient within an autonomous system to calculate a ContentObjectHash at a border router and carry it via trusted means within the autonomous system. If a Content Object ValidationAlgorithm does not have a KeyId then the Content Object cannot match an Interest with a KeyIdRestriction.

9. Request/Response Protocol

As an Interest moves through the network following the FIB table based on longest matching prefix, it leaves state at each forwarding node. The state is represented in a notional Pending Interest Table (PIT). The PIT tracks the Name, KeyIdRestriction, and ContentObjectHashRestriction to be matched by a Content Object. If a second Interest arrives with the same Name and selector values, it may be aggregated with the existing pending Interest. If the second Interest extends the Lifetime of the pending Interest, it should be forwarded to extend the life of downstream Interests.

When a Content Object arrives at a Forwarder, it is matched against the Interests in the PIT. For each matching Interest, the Content Object is forwarded along the reverse path of that PIT entry and the PIT entry is removed.

A Content Object that does not match any Interest in the PIT is dropped.

A Forwarder may implement a Content Object cache called a Content Store. At the core protocol level, the Content Store must obey similar rules as the Forwarder. If an Interest specifies a ContentObjectHashRestriction, the Content Store SHOULD NOT respond unless it has verified the hash of the Content Object. If the Interest carries a KeyIdRestriction then the Content Store MUST cryptographically verify the signature or not respond.

TODO: Specify what to do in case of failure.

10. Interest Return

This section describes the process whereby a network element may return an Interest message to a previous hop if there is an error processing the Interest. The returned Interest may be further processed at the previous hop or returned towards the Interest origin. When a node returns an Interest it indicates that the previous hop should not expect a response from that node for the Interest -- i.e. there is no PIT entry left at the returning node.

The returned message maintains compatibility with the existing TLV packet format (a fixed header, optional hop-by-hop headers, and the CCNx message body). The returned Interest packet is modified in only two ways:

- The PacketType is set to InterestReturn to indicate a Feedback message.

- The ReturnCode is set to the appropriate value to signal the reason for the return

The specific encodings of the Interest Return are specified in [CCNMessages].

A Forwarder is not required to send any Interest Return messages.

A Forwarder is not required to process any received Interest Return message. If a Forwarder does not process Interest Return messages, it should silently drop them.

The Interest Return message does not apply to a Content Object or any other message type.

An Interest Return message is a 1-hop message between peers. It is not propagated multiple hops via the FIB. An intermedate node that receives an InterestReturn may take corrective actions or may propagage its own InterestReturn to previous hops as indicated in the reverse path of a PIT entry.

10.1. Message Format

The Interest Return message looks exactly like the original Interest message with the exception of the two modifications mentioned above. The PacketType is set to indicate the message is an InterestReturn and the reserved byte in the Interest header is used as a Return Code. The numeric values for the PacketType and ReturnCodes are in [CCNMessages].

10.2. ReturnCode Types

This section defines the InterestReturn ReturnCode introduced in this RFC. The numeric values used in the packet are defined in [CCNMessages].

Name	Description
No Route (Section 10.3.1)	The returning Forwarder has no route to the Interest name.
HopLimit Exceeded (Section 10.3.2)	The HopLimit has decremented to 0 and need to forward the packet.
Interest MTU too large (Section 10.3.3)	The Interest's MTU does not conform to the required minimum and would require fragmentation.
No Resources (Section 10.3.4)	The node does not have the resources to process the Interest.
Path error (Section 10.3.5)	There was a transmission error when forwarding the Interest along a route (a transient error).
Prohibited (Section 10.3.6)	An administrative setting prohibits processing this Interest.
Congestion (Section 10.3.7)	The Interest was dropped due to congestion (a transient error).

Table 2: Interest Return Reason Codes

10.3. Interest Return Protocol

This section describes the Forwarder behavior for the various Reason codes for Interest Return. A Forwarder is not required to generate any of the codes, but if it does, it must conform to this specification.

If a Forwarder receives an Interest Return, it SHOULD take these standard corrective actions. A forwarder is allowed to ignore Interest Return messages, in which case its PIT entry would go through normal timeout processes.

- Verify that the Interest Return came from a next-hop to which it actually sent the Interest.

- If a PIT entry for the corresponding Interest does not exist, the Forwarder should ignore the Interest Return.

- If a PIT entry for the corresponding Interest does exist, the Forwarder MAY do one of the following:

  * Try a different forwarding path, if one exists, and discard the Interest Return, or

  * Clear the PIT state and send an Interest Return along the reverse path.

If a forwarder tries alternate routes, it MUST ensure that it does not use same same path multiple times. For example, it could keep track of which next hops it has tried and not re-use them.

If a forwarder tries an alternate route, it may receive a second InterestReturn, possibly of a different type than the first InterestReturn. For example, node A sends an Interest to node B, which sends a No Route return. Node A then tries node C, which sends a Prohibited. Node A should choose what it thinks is the appropriate code to send back to its previous hop

If a forwarder tries an alternate route, it should decrement the Interest Lifetime to account for the time spent thus far processing the Interest.

10.3.1. No Route

If a Forwarder receives an Interest for which it has no route, or for which the only route is back towards the system that sent the Interest, the Forwarder SHOULD generate a "No Route" Interest Return message.

How a forwarder manages the FIB table when it receives a No Route message is implementation dependent. In general, receiving a No Route Interest Return should not cause a forwarder to remove a route. The dynamic routing protocol that installed the route should correct the route or the administrator who created a static route should correct the configuration. A forwarder could suppress using that next hop for some period of time.

10.3.2. HopLimit Exceeded

A Forwarder MAY choose to send HopLimit Exceeded messages when it receives an Interest that must be forwarded off system and the HopLimit is 0.

10.3.3. Interest MTU Too Large

If a Forwarder receives an Interest whose MTU exceeds the prescribed minimum, it MAY send an "Interest MTU Too Large" message, or it may silently discard the Interest.

If a Forwarder receives an "Interest MTU Too Large" is SHOULD NOT try alternate paths. It SHOULD propagate the Interest Return to its previous hops.

10.3.4. No Resources

If a Forwarder receives an Interest and it cannot process the Interest due to lack of resources, it MAY send an InterestReturn. A lack of resources could be the PIT table is too large, or some other capacity limit.

10.3.5. Path Error

If a forwarder detects an error forwarding an Interest, such as over a reliable link, it MAY send a Path Error Interest Return indicating that it was not able to send or repair a forwarding error.

10.3.6. Prohibited

A forwarder may have administrative policies, such as access control lists, that prohibit receiving or forwarding an Interest. If a forwarder discards an Interest due to a policy, it MAY send a Prohibited InterestReturn to the previous hop. For example, if there is an ACL that says /parc/private can only come from interface e0, but the Forwarder receives one from e1, teh Forwarder must have a way to return the Interet with an explanation.

10.3.7. Congestion

If a forwarder discards an Interest due to congestion, it MAY send a Congestion InterestReturn to the previous hop.

11. Acknowledgements

12. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see Guidelines for Writing an IANA Considerations Section in RFCs [RFC5226] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

13. Security Considerations

The Interest Return message has no authenticator from the previous hop. Therefore, the payload of the Interest Return should only be used locally to match an Interest. A node should never forward that Interest payload as an Interest. It should also verify that it send the Interest in the Interest Return to that node and not allow anyone to negate Interest messages.

If two peers require authenticated messaging, they must use an external mechanism.

14. References

14.1. Normative References

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

14.2. Informative References

[CCN] PARC, Inc., "CCNx Open Source", 2007, <http://www.CCNx.org>.

[CCNMessages] Mosko, M., Solis, I., and M. Stapp, "CCNx Messages in TLV Format (Internet draft)", 2015, <http://tools.ietf.org/html/draft-mosko-icnrg-ccnxmessages-00>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

Authors' Addresses

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com

Ignacio Solis
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com