Workgroup: ICNRG
Internet-Draft: draft-irtf-icnrg-reflexive-forwarding-01
Updates: 8569, 8609 (if approved)
Published: 12 March 2025
Intended Status: Experimental
Expires: 13 September 2025
Authors:
D. Oran, *Network Systems Research and Design*
D. Kutscher, *HKUST(GZ)*
H. Asaeda, *NICT*
K. Calvert, *University of Kentucky*

Reflexive Forwarding for CCNx and NDN Protocols

Abstract

Current Information-Centric Networking protocols such as CCNx and NDN have a wide range of useful applications in content retrieval and other scenarios that depend only on a robust two-way exchange in the form of a request and response (represented by an *Interest-Data exchange* in the case of the two protocols noted above). A number of important applications however, require placing large amounts of data in the Interest message, and/or more than one two-way handshake. While these can be accomplished using independent Interest-Data exchanges by reversing the roles of consumer and producer, such approaches can be both clumsy for applications and problematic from a state management, congestion control, or security standpoint. This specification proposes a *Reflexive Forwarding* extension to the CCNx and NDN protocol architectures that eliminates the problems inherent in using independent Interest-Data exchanges for such applications. It updates RFC8569 and RFC8609.

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 13 September 2025.

Copyright Notice

Table of Contents

1. Introduction

Current ICN protocols such as CCNx [RFC8569] and NDN [NDN] have a wide range of useful applications in content retrieval and other scenarios that depend only on a robust two-way exchange in the form of a request and response. These ICN architectures use the terms "consumer" and "producer" for the respective roles of the requester and the responder, and the protocols directly capture the mechanics of the two-way exchange through the "Interest message" carrying the request, and the "Data message" carrying the response. Through these constructs, the protocols are heavily biased toward a pure *pull-based* interaction model where requests are small (carrying little or no user-supplied data other than the name of the requested data object), and responses are relatively large - up to an architecture-defined maximum transmission unit (MTU) on the order of kilobytes or tens of kilobytes.

A number of important applications however require interaction models more complex than individual request/response interactions in the same direction (i.e. between the same consumer and one or more producers). Among these we identify three important classes which are the target of the proposed enhancements defined in this specification. These are described in the following paragraphs.

Remote Method Invocation (RMI, aka RPC): When invoking a remote method, it is common for the method to require arguments supplied by the caller. In conventional TCP/IP style protocols like CORBA or HTTP "Post", these are pushed to the server as part of the message or messages that comprise the request. In ICN-style protocols there is an unattractive choice between inflating the request initiation with pushed arguments, or arranging to have one or more independent request/response pairs in the opposite direction for the server to fetch the arguments. Both of these approaches have substantial disadvantages. Recently, a viable alternative emerged through the work on RICE [Krol2018] which pioneered the main design elements proposed in this specification.

Phone-Home scenario: Applications in sensing, Internet-of-things (IoT) and other types where data is produced unpredictably and needs to be *pushed* somewhere create a conundrum for the pure pull-based architectures considered here. If instead one eschews relaxing the size

asymmetry between requests and responses, some additional protocol machinery is needed. Earlier efforts in the ICN community have recognized this issue and designed methods to provoke a cooperating element to issue a request to return the data the originator desires to push, essentially "phoning home" to get the responder to fetch the data. One that has been explored to some extent is the *Interest-Interest-Data* exchange [Carzaniga2011], where an Interest is sent containing the desired request as encapsulated data. CCNx-1.0 Bidirectional Streams [Mosko2017] are also based on a scheme where an Interest is used to signal a name prefix that a consumer has registered for receiving Interests from a peer in a bidirectional streaming session.

Peer state synchronization: A large class of applications, typified by those built on top of reliable order-preserving transport protocols, require initial state synchronization between the peers. This is accomplished with a three-way (or longer) handshake, since employing a two-way handshake as provided in the existing NDN and CCNx protocols exposes a number of well-known hazards, such as *half-open connections*. When attempted for security-related operations such as key exchange, additional hazards such as *man-in-the-middle* attacks become trivial to mount. Existing alternatives, similar to those used in the two examples above, instead utilize either overlapping Interest-Data exchanges in opposite directions (resulting in a four-way handshake) or by adding initialization data to the initial request and employing an Interest-Interest-Data protocol extension as noted in the Phone-home scenarios above.

All of the above application interaction models present interesting challenges, as neither relaxing the architecture to support pushing large amounts of data, nor introducing substantial complexities through multiple independent Interest-Data exchanges is an attractive approach. The following subsections provide further background and justification for why push and/or independent exchanges are problematical.

1.1. Problems with pushing data

There are two substantial problems with the simple approach of just allowing arbitrary amounts of data to be included with requests. These are:

1. In ICN protocols such as NDN and CCNx, Interest messages are intended to be small, on the order the size of a TCP ACK, as opposed to the size of a TCP data segment. This is because the hop-by-hop congestion control and forwarder state management requires Interest messages to be buffered in expectation of returning data, and possibly retransmitted hop-by-hop as opposed to end-to-end. In addition, the need to create and manage state on a per-Interest basis is substantially complicated if requests in Interest messages are larger than a Path MTU (PMTU) and need to be fragmented hop-by-hop.
2. If the payload data of a request is used for invoking a computation (as in the RMI case described above) then substantial bandwidth can be wasted if the computation is either refused or abandoned for any number of reasons, including the requestor failing an authorization check, or the responder not having sufficient resources to execute the associated computation.

These problems also exist in pure datagram transport protocols such as those used for legacy RMI applications like NFS [RFC7530]. More usual are application protocols like HTTP(s) which rely on the TCP or QUIC 3-way handshake to establish a session and then have congestion control and segmentation provided as part of the transport protocol, further allowing sessions to be rejected before large amounts of data are transmitted or significant computational resources expended.

1.2. Problems with utilizing independent exchanges

In order to either complete a three-way handshake, or fetch data via a pull from the original requestor, the role of consumer and producer need to be reversed and an Interest/Data exchange initiated in the direction opposite of the initiating exchange. When done with an independent Interest/Data request and response, a number of complications ensue. Among them are:

1. The originating consumer needs to have a routable name prefix that can be used for the exchange. This means the consumer must arrange to have its name prefix propagated in the ICN routing system with sufficient reach that the producer issuing the interest can be assured it is routed appropriately. While some consumers are generally online and act as application servers, justifying the maintenance of this routing information, many do not. Further, in mobile environments, a pure consumer that does not need to have a routable name prefix can benefit from the inherent consumer mobility support in the CCNx and NDN protocols. By requiring a routable name prefix, extra mobile routing machinery is needed, such as that proposed in KITE [Zhang2018] or MAPME [Auge2018].
2. The consumer name prefix in item (1) above must be communicated to the producer as a payload, name suffix, or other field of the initiating Interest message. Since this name in its entirety is chosen by the consumer, it is highly problematic from a security standpoint, as it can recruit the producer to mount a reflection attack against the consumer's chosen victim.
3. The correlation between the exchanges in opposite directions must be maintained by both the consumer and the producer as independent state, as opposed to being architecturally tied together as would be the case with a conventional 3-way handshake finite state machine. While this can of course be accomplished with care by both parties, experience has shown that it is error prone (for example see the checkered history of interactions between the SIP [RFC3261] and SDP Offer-Answer [RFC6337]) protocols. When employed as the wrapper for a key management protocol such as with TLS [RFC8446] state management errors can be catastrophic for security.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Definitions

For general ICN-related terms, we adopt the terms defined in [RFC8793]. For CCNx-specific terms, we use the terms defined in [RFC8569] and [RFC8609]. This specification defines the following additional terms:

Reflexive Interest (RI) An interest message sent from a producer back towards a consumer to fetch data needed to satisfy the original interest.

Trigger Interest (TI) An interest message sent from a consumer to a producer to potentially trigger one or more Reflexive Interests sent by the producer back to the consumer.

Reflexive Name Segment A typed name segment which, when present as the initial (highest-order) name segment identifies an Interest as being a Reflexive Interest instead of a normal Interest.

Reflexive Name Prefix (RNP) A value carried in an Interest message to indicate to a producer that it may fetch data from the sending consumer by issuing one or more corresponding Reflexive Interests. It consists of a Name prefix whose high-order (i.e. first) name segment is a Reflexive Name Segment as defined above. It acts as a nonce, distinguishing Reflexive Interests related to a specific Trigger Interest from others in the same space-time vicinity of the network.

> **Note:** the above definition may need to be adjusted depending on the decision about protocol encoding discussed in Section 4.3.1.

3. Overview of the Reflexive Forwarding design

This specification defines a *Reflexive Forwarding* extension to CCNx and NDN that avoids the problems enumerated in Sections 1.1 and 1.2. It straightforwardly exploits the hop-by-hop state and path symmetry properties of the current protocols.

Figure 1 below illustrates a canonical NDN/CCNx forwarder with its conceptual data structures of the Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB). The key observation involves the relation between the PIT and the FIB. Upon arrival of an Interest, a PIT entry is created which contains state recording the incoming interface on which the Interest was received. If the Interest is not immediately satisfied by cached data in the CS, the forwarder looks up the name in the FIB to ascertain the *next-hop* to propagate the Interest onward upstream toward the named producer. Therefore, a chain of forwarding state is established during Interest forwarding that couples the PIT entries of the chain of forwarders together conceptually as *breadcrumbs*. These are used to forward the returning Data Message over the reverse path through the chain of forwarders until the Data message arrives at the originating consumer. The state in the PITs is *unwound* by destroying it as each PIT entry is *satisfied*. This behavior is **critical** to the feasibility of the reflexive forwarding design we propose.

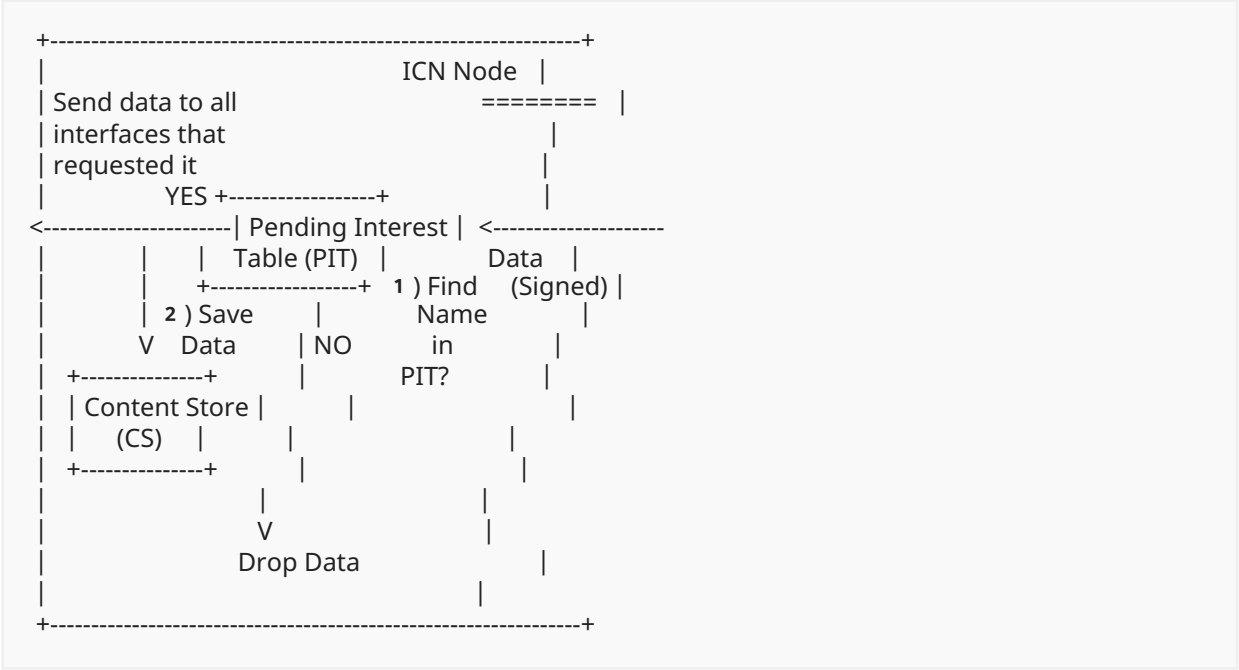


Figure 1: ICN forwarder structure

Given the above forwarding properties for Interests, it should be clear that while an Interest is outstanding and ultimately arrives at a producer who can respond to it, there is sufficient state in the chain of forwarders to route not just a returning Data message, but potentially another Interest directed through the reverse path to the unique consumer who issued the original Interest. (Section 6.1.3 describes how Interest aggregation of requests to the same target name from multiple consumers interacts with this scheme.) The key question therefore is how to access this state in a way that it can be used to forward Interests from the producer back to the consumer.

In order to achieve this *Reflexive Interest* forwarding on the reverse path recorded in the PIT of each forwarder, we need a few critical design elements:

1. The Reflexive Interest needs to have a *Name*. This name is what the originating consumer will use to match against the Data object (or multiple Data objects — more on this later) that the producer may request by issuing the Reflexive Interest. This cannot be just any name, but needs to essentially name the state already recorded in the PIT and not allow the consumer to manufacture an arbitrary name and mount a reflection attack as pointed out in Section 1.2, Paragraph 2, Item 2.
2. Each forwarder along the reverse path from producer to consumer must be able to forward the Reflexive Interest towards the direction of the Consumer without relying on global routing information, as the Reflexive Name Prefixes are only valid while the originating Interest/Data exchange state is present at the forwarders. Essential to this operation is the ability to access state created by the PIT entry associated with the Trigger Interest message since that is the state necessary to identify the ingress face of the Trigger Interest. This is the unique output face (modulo interest aggregation) over which the Reflexive Interest needs to be forwarded. The Name assigned by the consumer for Reflexive Name Prefix in theory is adequate to the task, but entails a potentially expensive and complicated lookup procedure.
3. There has to be coupling of the state between the originating Interest-Data exchange and the enclosed Reflexive Interest-Data exchange at both the consumer and the producer. In our design, this is accomplished by the way Reflexive Interest names are chosen.

4. Detailed Specification of Reflexive Forwarding

The following sections provide the normative details on each of these design elements.

4.1. Generic Forwarder Design

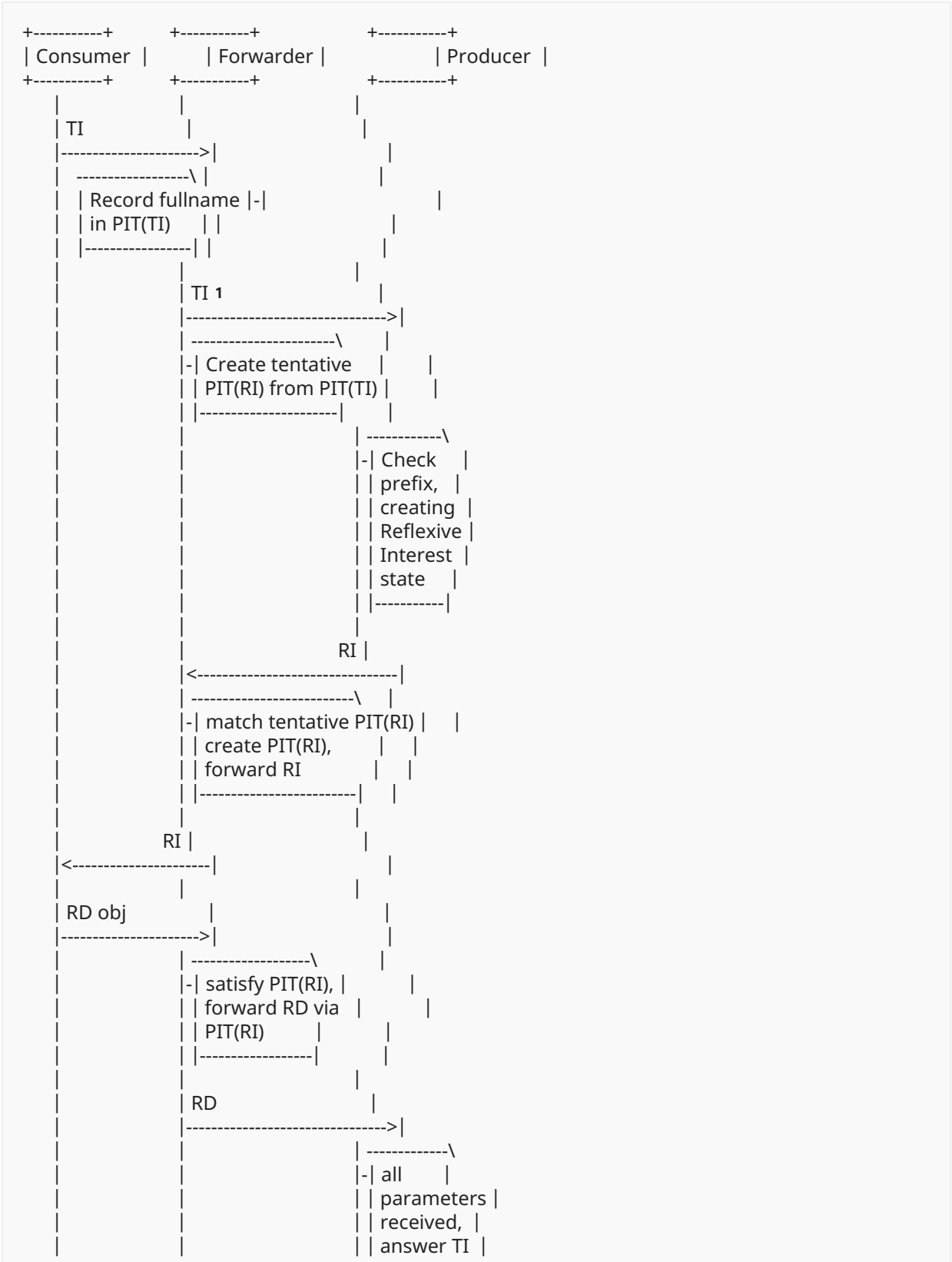
The following describes the operation of a CCNx or NDN forwarder enhanced to accommodate reflexive forwarding via Trigger Interests, Reflexive Interests, and Reflexive Data messages. This description likely adequate for most implementation approaches. However, there are possible performance bottlenecks for reflexive forwarding that employ highly parallel forwarding schemes using multi-core server systems. Such systems may use *PIT Sharding* across cores, and therefore benefit from additional optional protocol enhancements, such as the "PIT Token" protocol elements pioneered in [Shi2020]. Alternatively, techniques such as those employed by the Cefore [Asaeda2019][Cefore] forwarder from NICT may be employed.

4.2. Interaction Flow for Reflexive Forwarding: An Example

Based on the implementation approach in Cefore [Asaeda2019][Cefore], an example of the overall interaction flow for reflexive forwarding for CCNx is presented below and illustrated in Figure 2.

Forwarder operation for CCNx is enhanced in the following respects when supporting Reflexive Interests.

- When a forwarder receives a Trigger Interest (TI) that will be sent toward the producer, in addition to the normal PIT entry created on receipt of a new Interest message, the forwarder creates a *tentative* PIT entry for the possible later receipt of a Reflexive Interest
- This tentative PIT entry has the Reflexive Name Prefix from the Trigger Interest as its Name field, and the incoming face list of the trigger interest as the potential output face(s) for any subsequent Reflexive Interest received.
- On receipt of a Reflexive Interest matching the Reflexive Name Prefix of a tentative PIT entry, a full PIT entry is created from the tentative entry and normal PIT processing ensues (e.g. also including recording the incoming face(s) of the Reflexive Interest).
- The Reflexive Interest is forwarded over the recorded outgoing face(s) from the PIT as opposed to examining the FIB to find an appropriate next hop for that Reflexive Interest.


Consumer
Forwarder
Producer
TI
Record fullname in PIT(TI)
TI 1
Create tentative PIT(RI) from PIT(TI)
Check prefix, creating Reflexive Interest state
RI
match tentative PIT(RI)
create PIT(RI), forward RI
RI
RD obj
satisfy PIT(RI), forward RD via PIT(RI)
RD
all parameters received, answer TI

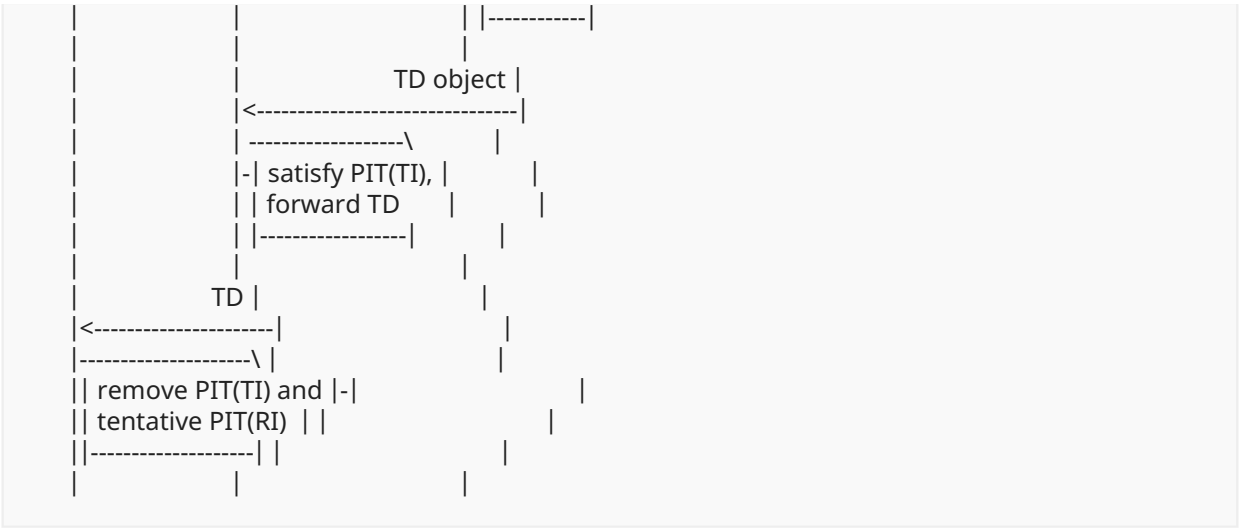


Figure 2: Overview of Reflexive Forwarding in CCNx

There are a number of reasons to maintain both a tentative RI-PIT and (possibly) multiple RI-PIT entries for reflexive forwarding. Among them are:

- There may be multiple chunks [I-D.irtf-icnrg-ccnxchunking] requested using a sequence of Reflexive Interests.
- Multiple Reflexive Interests may be issued by the producer to fetch arguments in the remote invocation use case described in Section 5.1.

This is further discussed in Section 4.3.2 on how to handle multiple reflexive interests sent in the context of a single Trigger Interest / Trigger Data interaction.

4.3. Naming of Reflexive Interests

A consumer may have one or more objects for the producer to fetch, and therefore needs to communicate enough information in its initial Trigger Interest to allow the producer to construct properly formed Reflexive Interest names. For some applications the set of *full names* (see the ICN Terminology RFC [RFC8793]) is known a priori, for example through compile time bindings of arguments in an interface definition or by the architectural definition of a simple sensor reading. In other cases, the full names of the individual objects must be communicated (or inferred) via the Reflexive Name Prefix (RNP) in the Trigger Interest message.

We define a new name segment type, the *Reflexive Name Segment* which holds the *Reflexive Name Prefix* in Reflexive Interest and Reflexive Data messages. The Reflexive Name Segment MUST be the first (i.e. high order) name segment of any Reflexive Interest issued by a producer. In CCNx, the Reflexive Name Segment type value will be registered in the IANA registry for [RFC8609] when approved (see Section 9).

The value of any Reflexive Name Prefix is assigned by the consumer. The selected value MUST provide sufficient entropy to uniquely identify the issuing consumer for the duration of any outstanding Trigger Interest-Trigger Data exchange. Any scheme that provides sufficient randomness and entropy can suffice, but the consumer SHOULD choose a different random value for each Reflexive Name Prefix it constructs because:

1. A collision of Reflexive Name Prefixes at any of the intermediate forwarders would result in the same mutability problems generated by poor name selection in other contexts, such as mis-routing of Reflexive Interests; and
2. Re-use of the same reflexive interest name over multiple interactions might reveal linkability information that could be used by surveillance adversaries for tracking purposes.

A UUID, which is a randomly generated 128 bit quantity specified in [RFC9562] can be used for the value of the Reflexive Name Segment.

This initial name segment in the Reflexive Interest is prepended to any object names the consumer wishes the producer to fetch. More than one object may be needed by the producer for the current Interest-Data interaction. There are three cases to consider:

1. The reflexive *fullname* of a single object to fetch.
2. A single reflexive name prefix out of which the producer can (by application-specific means) construct a number of *fullnames* of the objects it may want to fetch.
3. The reflexive *fullname* of a FLIC Manifest [I-D.irtf-icnrg-flic] enumerating the suffixes that may be used by the producer to construct the necessary names. We distinguish this from the single object fetch in case (1) above because the use of a Manifest implies multiple Reflexive Interest/Data exchanges with the consumer.

A producer, upon receiving a Trigger Interest with a Reflexive Name Prefix, may decide it needs to retrieve the associated data object(s). It therefore can issue one or more Reflexive Interests by appending the necessary name segments needed to form valid full names of the associated objects present at the originating consumer. These in fact comprise conventional Interest-Data exchanges, with no alteration of the usual semantics with regard to signatures, caching, expiration, etc. When the producer has retrieved the required objects to complete the original Interest-Data exchange, it can issue its Data response, which unwinds all the established state at the producer, the consumer, and the intermediate forwarders.

4.3.1. How to represent/encode the Reflexive Name Prefix in the Trigger Interest

While the design is firm on how to represent the Reflexive Name Prefix (RNP) in Reflexive Interests (and, by extension in the corresponding Reflexive Data response) by using a Reflexive Name Segment as the highedt-order name segment of the Reflexive Interest, we have studied two reasonable approaches to how the RNP should be represented in Trigger Interests. These are:

1. as a Reflexive Name Segment of the Trigger Interest Name. This is the same registered name segment type as used for Reflexive Interests, but present as the last (i.e. *trailing*) name segment.
2. as a separately defined and IANA registered message-level TLV in the Trigger Interest.

There are tradeoffs in the choice of encoding. We outline the ones we are aware of below. Further experimentation with implementing the protocol and using it will inform a final decision on which approach to codify.

> **Note:** Approach 1 is currently the method implemented by the Cefore forwarder [Cefore]

- Approach 1 likely limits the ability to make use of caching for Trigger Data responses. This is because no two TIs from different consumers would carry the same name - at least within the lifetime of the TI-TD pair. It is however unclear whether that is a significant problem. In many, perhaps most, applications where RI is needed (e.g., remote method invocation, access to paywalled content) the returned Trigger Data is unlikely to be usefully cacheable (e.g., because it is encrypted with a subscriber-specific key/watermark).
- Approach 2 requires a separate check to avoid incorrect interest aggregation, since trigger interests with different RNP TLVs should not be aggregated. Approach 1 “just works” since the names are different and won’t be aggregated. However, this may be of little consequence since interest aggregation checks already require a forwarder to examine other optional Interest message fields such as *Interest Payload*. See [RFC8609] Section 2.4.2 for more information.
- Approach 1 doesn’t have to echo back the value in the finishing Trigger Data response, making it smaller. That might matter a bit in the sensor phone-home use case (Section 5.3), but it’s likely not a big deal given that RNPs allocated according to the UUID scheme are only 128 bits and hence unlikely to make that much of a difference.

4.3.2. Naming of multiple Reflexive Interests

As noted in the general description of Section 4.2, a given use case may require multiple RI/RD exchanges within the scope of a single TI/TD exchange.

One example is an object to be returned via reflexive data that requires multiple chunks. Following are some examples of the names without/with chunk information enclosed in TI and RIs:

```
TI: ccnx:/Name=example.com/RNP=RNP 1
RI: ccnx:/RNP=RNP 1

TI: ccnx:/Name=example.com/RNP=RNP 1
RI_ 1 : ccnx:/RNP=RNP 1 /Chunk= 0   ---
RI_ 2 : ccnx:/RNP=RNP 1 /Chunk= 1   EndChunkNumber= 2
RI_ 3 : ccnx:/RNP=RNP 1 /Chunk= 2   EndChunkNumber= 2
```

Figure 3: Responses without/with multiple chunks using Reflexive Interests

The RI-PIT entry MAY include a chunk number to the RNP, while a tentative RI-PIT entry does not. Like an ordinary PIT entry, RI-PIT is erased whenever the corresponding RD is forwarded through the downstream face, while a tentative RI-PIT entry is erased when the TI-PIT lifetime expires or Trigger Data (TD) is received and forwarded at the forwarder.

> **Note:** The above likely needs some adjustment when we sort out the general handling on chunking for both regular and reflexive interests. Stay tuned.

A second common use case is argument fetch for remote method invocation.

> **Note:** An example will be included in a later draft version

4.3.3. Recursive Reflexive Interest invocation

There is no explicit prohibition in this version of the specification to use a Reflexive Interest as a Trigger Interest to recursively create a multi-way handshake protocol. It is not yet clear if there are any compelling use cases for this or if there are undiscovered hazards by permitting it. However, we are confident that we do have sufficient protections against resource depletion attacks that could be attempted through this technique.

4.4. Consumer Operation

A consumer wishing to employ Reflexive Forwarding MUST include a Reflexive Name Prefix (RNP) in the Interest, which causes it to be treated as a Trigger Interest by Forwarders and Producers. The producer can subsequently craft Reflexive Interests with names using the RNP as a Reflexive Name Segment. Upon receiving a Reflexive Interest (e.g. RI in Figure 2) from a producer in response to the Interest whose first name segment is the RNP supplied earlier, the consumer SHOULD perform a full name match against the object specified in the RI, and return that object to the producer in a conventional Data message, (e.g. RD in Figure 2).

4.5. Producer Operation

A producer that has received an Trigger Interest with a Reflexive Name Prefix (RNP) keeps the supplied RNP from the Trigger Interest for subsequent (optional, depending on application semantics) Reflexive Interest sending.

When sending a Reflexive Interest back to the consumer, the producer MUST construct a corresponding Interest name based on the RNP in the reflexive Interest.

4.6. Forwarder Operation

The forwarder performs the following operations.

1. Upon receiving an Interest containing a Reflexive Name Segment as its highest-order name segment, the Interest is interpreted as a Reflexive Interest. A forwarder MAY check its CS for a matching Data Object. If a match is found, the corresponding Data is returned and the Reflexive Interest is considered satisfied. If no match is found, proceed with the following steps.

   Note: Given that reflexive interests initiate exchanges constrained by the Interest Lifetime of the enclosing Trigger Interest/Data exchange, the value of caching the Reflexive Data responses in forwarders is for short-term recovery from lost packets as opposed to optimizing longer term caching gain. For this reason we avoid mandating or strongly recommending the CS check for reflexive interests.

2. The forwarder MUST check for the existence of a matching Tentative PIT entry (i.e., one containing the RNP in this Interest's Name). If no matching Tentative PIT entry is found, it could, strictly speaking, be considered an error, but the forwarder SHOULD simply process the Interest as a normal non-reflexive Interest and skip the steps below. A match indicates that this is a Reflexive Interest corresponding to the original Trigger Interest, so execute the following steps.
3. Create a new PIT entry for the Reflexive Interest (if resources are sufficient). In the case of [Cefore] this is accomplished by cloning the new PIT entry from the tentative PIT entry created earlier by receipt of the Trigger Interest. (For interactions with Interest aggregation, also see Section 6.1.3)
4. Look up the ingress face from the originating Trigger Interest's PIT entry, and forward the Reflexive Interest on this face. In the case of [Cefore] the face to forward on would have already been populated in the Reflexive Interest's PIT entry.

The PIT entry for the Reflexive Interest is consumed per regular Interest/Data message forwarding requirements. The PIT entry for the originating Interest (that communicated the Reflexive Interest Name) is also consumed by a final Data message from the producer to the original consumer.

4.6.1. Forwarder algorithms in pseudocode

This section provides some pseudocode examples to further explain the details of forwarder operation. It has separate code paths for minimal forwarder operations and those needed by high-performance forwarders as is further discussed in Section 6.1.1, Paragraph 4.

4.6.1.1. Processing of a Trigger Interest containing a Reflexive Name Prefix

```
Create PIT entry for Interest;
Optionally create a tentative PIT entry for any potential Reflexive Interests
    expected to match the RNP in the Trigger Interest;
Forward Interest upstream;
```

4.6.1.2. Processing of a Reflexive Interest

```
Use Reflexive Name Segment of Reflexive Interest's Name to look up the Trigger Interest PIT entry;

IF PIT entry of original Interest not is found
   Issue an Interest Return with "No Route" error
       back to the producer;
ELSE
   Create PIT entry for Reflexive Interest
   (or optionally clone it from the tentative TI PIT entry that was created together
    with the Trigger Interest PIT entry);
   Process as a normal Interest;
RETURN
```

4.7. State coupling between producer and consumer

A consumer that wishes to use this scheme MUST utilize one of the reflexive naming options defined in Section 4.3 and include it in the corresponding Interest message. The Reflexive Name Prefix *and* the full name of the requested data object (that identifies the producer) identify the common state shared by the consumer and the producer. When the producer responds by sending Interests with a Reflexive Name Prefix, the original consumer therefore has sufficient information to map these Interests to the ongoing Interest-Data exchange.

The exchange is finished when the producer who received the Trigger Interest message responds with a Data message (or an Interest Return message in the case of error) answering the Trigger Interest. After sending this Data message, the producer SHOULD destroy the corresponding shared state. It MAY decide to use a timer that will trigger a later state destruction. After receiving this Data message, the originating consumer MUST destroy the corresponding Interest-Data exchange state.

5. Use cases for Reflexive Interests

5.1. Achieving Remote Method Invocation with Reflexive Interests

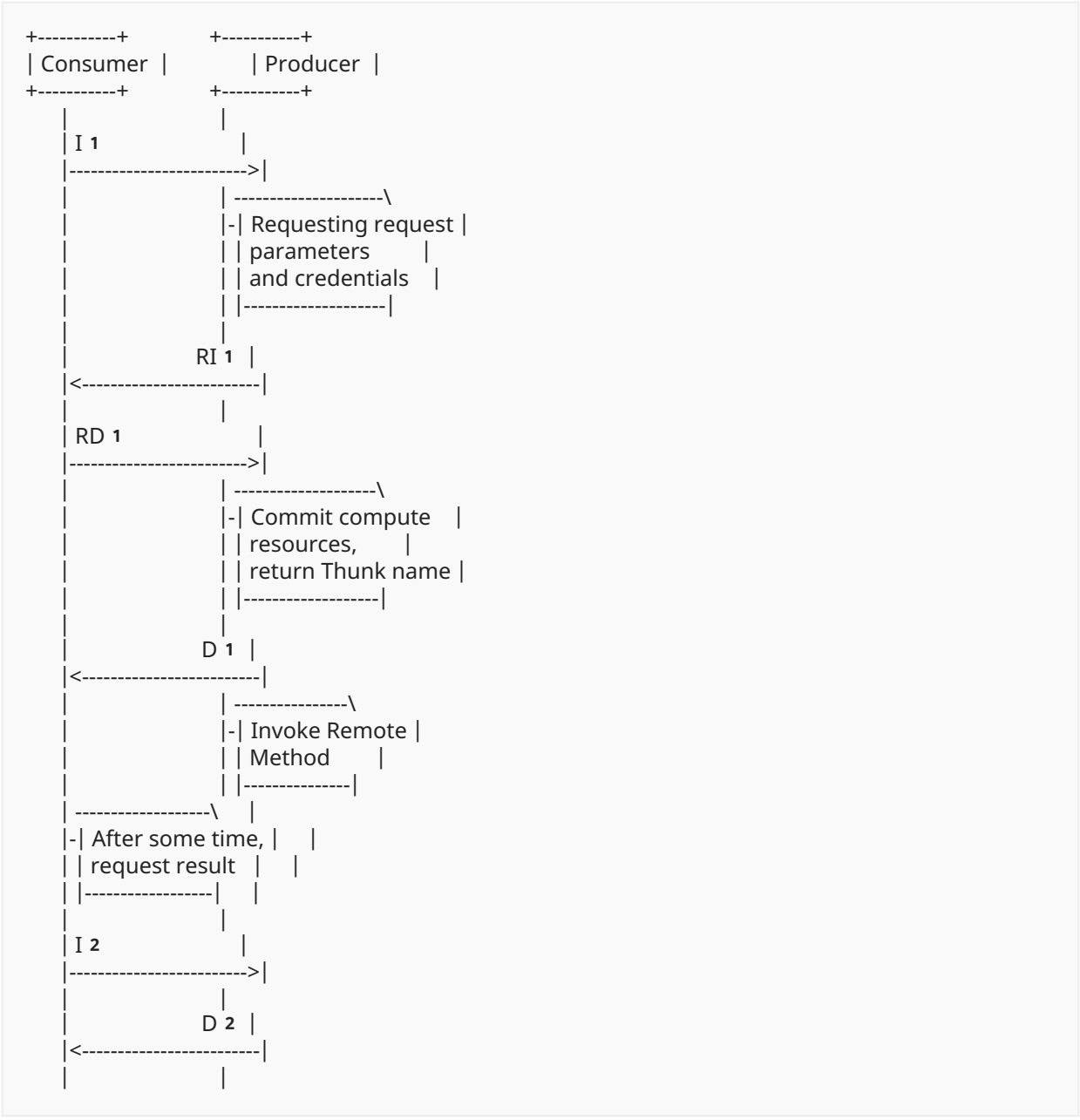
RICE (Remote Method Invocation in ICN) [Krol2018] used a similar Reflexive Interest Forwarding scheme that inspired the design specified in this document (similar to the original design captured in Appendix A.1).

In RICE, the original Interest denotes the remote method (plus potential parameters) to be invoked at a producer (server). Before committing any computing resources, the server can then request authentication credentials and (optional) parameters using reflexive Interest-Data exchanges.

When the server has obtained the necessary credentials and input parameters, it can decide to commit computing resources, start the compute process, and return a handle ("Thunk") in the final Data message to the original consumer (client).

The client would later request the computation results using a regular Interest-Data exchange (outside the Reflexive-Interest transaction), using the Thunk as a name for the computation result.

Figure 4 depicts an abstract message diagram for RICE. In addition to the 4-way Reflexive Forwarding Handshake for the details of the interaction, RICE adds another (standard) ICN Interest/Data exchange for transmitting the RMI result. The Thunk name is provided to the consumer in the D1 DATA message (answering the initial I1 Interest).


Consumer
Producer
I 1
Requesting request parameters and credentials
RI 1
RD 1
Commit compute resources, return Thunk name
D 1
Invoke Remote Method
After some time, request result
I 2
D 2

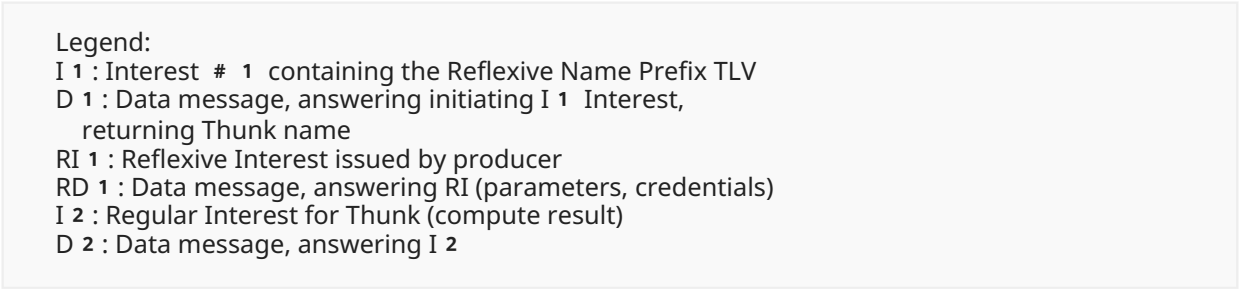


Figure 4: RICE Message Flow

5.2. RESTful Web Interactions

In todays HTTP-based web, RESTful (Representational State Transfer) web interactions are realized by sending requests in a client/server interaction, where the requests provides the application context (or a reference to it). It has been noted in [Moiseenko2014] that corresponding requests often exceed the response messages in size, and that this raises the problems noted in Section 1.1 when attempting to map such exchanges directly to CCNx/NDN.

Another reason not to include all request parameters in a (possibly encrypted) Interest message is the fact that a server (that is serving thousands of clients) would be obliged to receive, possibly decrypt and parse the complete requests before being able to determine whether the requestor is authorized, whether the request can be served etc. Many non-trivial requests could thus lead to computational overload attacks.

Using Reflexive Interest Forwarding for RESTful Web Interactions would encode the REST request in the original request, together with a Reflexive Interest Prefix that the server could then use to get back to the client for authentication credentials and request parameters, such as cookies. The request result (response message) could either be transmitted in the Data message answering the original request, or — in case of dynamic, longer-running computations — in a seperate Interest/Data exchange, potentially leveraging the Thunk scheme described in Section 5.1.

Unlike approaches where clients have to signal a globally routable prefix to the network, this approach would not require the client (original consumer) to expose its identity to the network (the network only sees the temporary Reflexive Name Prefix), but it would still be possible to authenticate the client at the server.

An initial design for achieving RESTful ICN that employs reflexive forwarding is presented in [Kutscher2022].

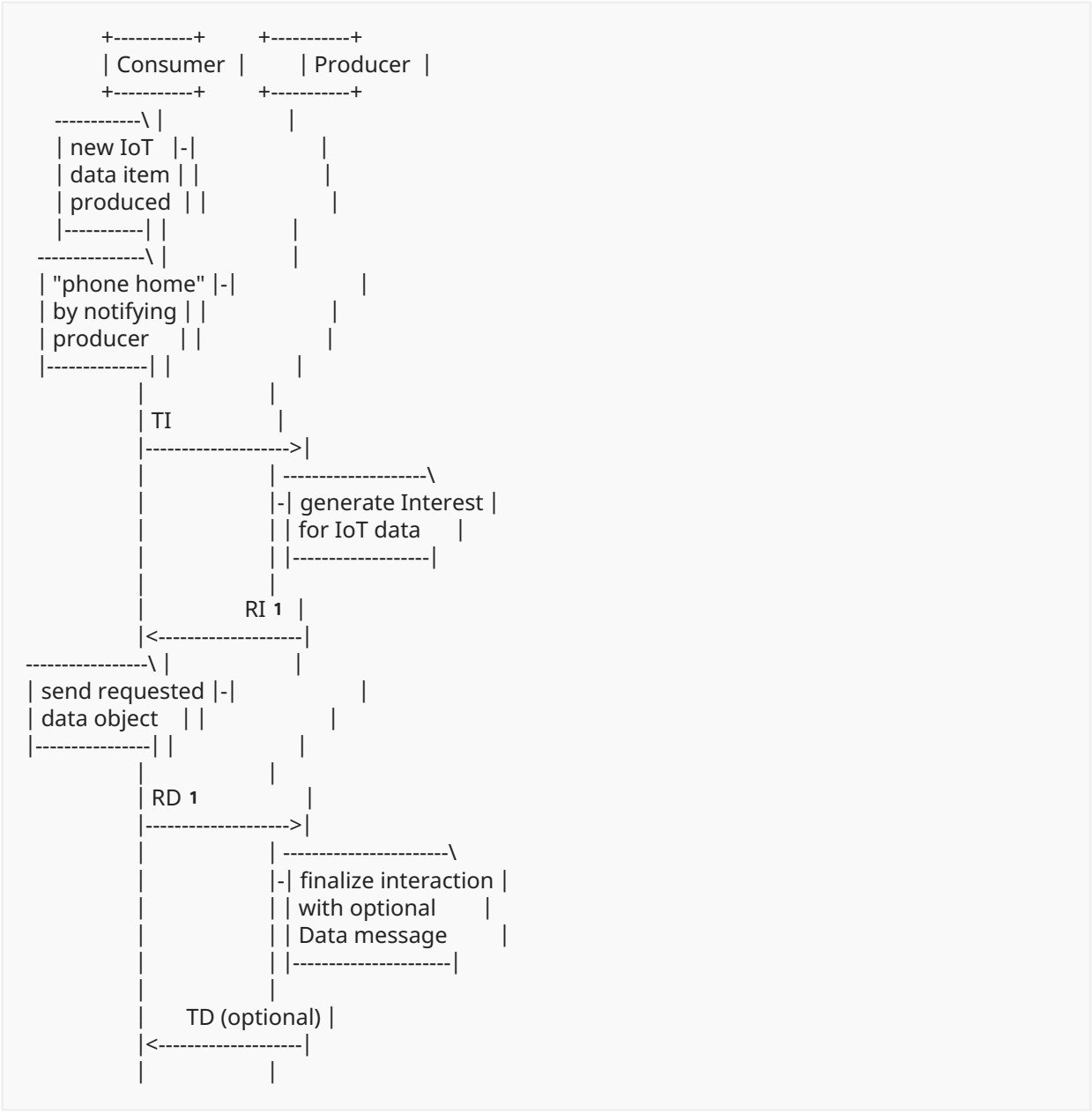
5.3. Achieving simple data pull from consumers with reflexive Interests

An oft-cited use case for ICN network architectures is *Internet of Things* (IoT), where the sources of data are limited-resource sensor/actuators. Many approaches have been tried (e.g. [Baccelli2014], [Lindgren2016], [Gundogan2018]) with varying degrees of success in addressing

the issues outlined in Section 1.1. The reflexive forwarding extension may substantially ameliorate the documented difficulties by allowing a different model for the basic interaction of sensors with the ICN network.

Instead of simply acting as a producer (either directly to the Internet or indirectly through the use of some form of application-layer gateway), the IoT device need only act as a consumer to initiate commuhication. When it has data to provide, it issues a "phone-home" Trigger Interest message to a pre-configured (or discovered) rendezvous name (e.g. an application-layer gateway or ICN Repo [Chen2015]) and provides a reflexive name prefix for the data it wishes to publish. The target producer may then issue the necessary Reflexive Interest message(s) to fetch the data. Once fetched, validated, and stored, the producer then responds to the Trigger Interest message with a success indication, possibly containing a Data object if needed to allow the originating device to modify its internal state. Alternatively, the producer might choose to not respond and allow the Trigger Interest to time out, although this is NOT RECOMMENDED except in cases where the extra message transmission bandwith is at a premium compared to the persistence of stale state in the forwarders. We note that this interaction approach mirrors the earlier efforts using Interest-Interest-Data designs.

Figure 5 depicts this interaction with the (optional) TD message.



Legend:
TI: Trigger Interest containing the Reflexive Name Prefix
RI 1 : Reflexive Interest requesting the IoT data
RD 1 : Data message, answering RI, returning IoT data object
TD: (optional) Data answering I 1

Figure 5: "Phone Home" Message Flow

There are two approaches that the IoT device can use for its response to a Reflexive Interest. It can simply construct a Data Message bound through the usual ICN hash name to the Reflexive Interest name. Since the scope of any data object bound in this way is only the duration of the enclosing Interest-Data exchange (see Section 6.2) the producer would need to itself construct any persistent Data object, name it, and sign it. This is sometimes the right approach, as for some applications the identity of the originating IoT device is not important from an operational or security point of view; in contrast the identity of the gateway or Repo is what matters.

If alternatively, the persistent Data object should be bound from a naming and security point of view to the originating IoT device, this can be easily accomplished. Instead of directly placing the content in a Data object responding to the reflexive Interest as above, the consumer encapsulates a complete CCNx/NDN Data message (which includes the desired name of the data) in the response to the Reflexive Interest message.

The interaction model described above brings a number potential advantages, some obvious, some less so. We enumerate a few of them as follows:

- By not requiring the IoT device to be actively listening for Interests, it can sleep and only wake up if it has something to communicate. Conversely, parties interested in obtaining data from the device do not need to be constantly polling in order to ascertain if there is new data available.
- No forwarder resources are tied up with state apart from the actual reflexive forwarding interactions. All that is needed is enough routing state in the FIB to be able to forward the "phone home" Interest to an appropriate target producer. While this model does not provide all the richness of a full Pub/Sub system (like that described in [Gundogan2018]) we argue it is adequate for a large subclass of such applications.
- The Reflexive interest, through either a name suffix or Interest payload, can give the IoT device useful context from which to craft its Data object in response. One highly useful parameter would be a robust clock value for the device to use as a timestamp of the data, possibly as part of its name, to correctly place it in a time seres of sensor readings. This substantially alleviates the need for low-end devices to have a robust time base, as long as they trust the producer they contact to provide it.

6. Implementation Considerations

There are a number of important aspects to the reflexive forwarding design which affect correctness and performance of existing forwarder, consumer, and producer implementations desiring to support it. This section discusses the effect of each of these elements on the protocol architecture.

6.1. Forwarder implementation considerations

6.1.1. Interactions with Input Processing of Interest and Data packets

Reflexive Interests are designed specifically to be no different from any other Interest other than the use of the Reflexive Name Segment type as their high-order name segment. This means that a forwarder does not have to have special handling in terms of creation, and destruction, and

other Interest processing needs such as timeouts, Interest satisfaction, and caching of returning data in the CS if desired. However, this design does require additional processing for Reflexive Interests not needed in the absence of reflexive forwarding. The most significant requirements are:

- In order to locate the matching RNP from the Trigger Interest, the forwarder's packet input processing needs to be able to efficiently access a PIT entry storing the RNP of the corresponding Trigger Interest.
- Ensure that the highest order name segment of the Reflexive Interest is a Reflexive Name Segment that matches the RNP stored in that PIT entry.

There are a few additional considerations to highlight for high-speed forwarders however; these are discussed in the following paragraphs.

In order to achieve forwarding scalability, high speed forwarders need to exploit available parallelism in both CPU (through multiple cores) and memory (through multiport DRAM) and limiting accesses to both DRAM and L3 caches). One commonly-used technique is *PIT sharding*, where the forwarder-global PIT is partitoned among cores such that all processing of both Interest and Data for a given Name is directed at the same core, optimizing both L1 I-cache utilization and L2/L3/DRAM throughput and latency. This is achieved in a number of implementations (e.g. [So2013]) by hashing the fullname in the Interest or Data and using that hash to select the assigned processing core (and associated memory banks). This efficiently distributes the load and minimizes the number of memory accesses other than to bytes of the input packet.

Straightforward input name hashing to achieve a sharded PIT has one potentially undesirable side effect: the Trigger Interest containing the Reflexive Name Prefix and any resultant Reflexive Interests issued by the producer will likely hash to different PIT shards, making any pointers that need to be traversed across shards or cross-shard updates expensive, possibly dramatically so. One could either optimize those accesses (as, for example, suggested in the discussion of Interest Lifetime in Section 6.1.2) or add special input handling of reflexive interests to steer them to the same shard as the original interest.

6.1.2. Interactions with Interest Lifetime

If and when a producer decides to fetch data from the consumer using one or more reflexive Interest-Data exchanges, the total latency for the original Interest-Data exchange is inflated, potentially by multiple RTTs. It is difficult for a consumer to predict the inflation factor when issuing the Trigger Interest, and hence there can be a substantial hazard of that Interest lifetime expiring before completion of the full multi-way exchange. This can result in persistent failures, which is obviously highly undesirable.

There is a fairly straightforward technique that can be employed by forwarders to avoid these "false" Interest lifetime expirations. In the absence of a superior alternative technique, it is RECOMMENDED that all forwarders implement the following algorithm.

If and when a Reflexive Interest arrives matching the Trigger Interest's RNP, the forwarder examines the Interest lifetime of the arriving Reflexive Interest. Call this value IL_r. The forwarder computes MAX(IL_t, (IL_r * 1.5)), and replaces IL_t with this value. This in effect ensures that the remaining Interest lifetime of the Trigger Interest accounts for the additional 1.5 RTTs that may occur as a result of the Reflexive Interest-Data exchange.

> We note that this is not unduly expensive in this design where the two PIT entries are guaranteed to be in the same PIT shard on a high speed forwarder. The earlier design discussed in Appendix A.1.2 required some additional gymnastics.

While the above approach of inflating the interest lifetime of the Trigger Interest to accommodate the additional RTTs of reflexive Interest-Data exchanges avoids the timeout problem, this does introduce a new vulnerability that must be dealt with. A Producer, either through a bug or malicious intent, could keep an originating Interest-Data exchange alive by continuing to send reflexive Interests back to the consumer, while the consumer had no way to terminate the enclosing interaction (there is no "cancel Interest" function in either NDN nor CCNx). To eliminate this hazard, if the consumer rejects a Reflexive Interest with a T_RETURN_PROHIBITED error, the forwarder(s), in addition to satisfying the corresponding PIT entry, MUST also delete the RNP from the Trigger Interest's PIT entry, thereby preventing any further Reflexive Interests from reaching the consumer. This in turn allows the enclosing Interest-Data exchange to either time out or be correctly ended with a Data message or Interest Return from the Producer.

6.1.3. Interactions with Interest aggregation and multi-path/multi-destination forwarding

As with numerous other situations where multiple Interests for the same named object arrive containing different parameters (e.g. Interest Lifetime, QoS, payload hash) the same phenomenon occurs for the Reflexive Name Prefix. If Trigger Interests with different Reflexive Name Prefixes collide, the forwarder MUST NOT aggregate these Interest messages and instead MUST create a separate PIT entry for each.

Forwarders supporting multi-path forwarding may of course exploit this capability for Trigger Interests with identical Reflexive Name Prefixes, like any other Interests. There are two sub-cases of multi-next hop behavior; regular multi-path (where the split traffic reconverges further upstream) and multi-destination (where it doesn't and the Interest reaches multiple producers).

For multi-path, since the Interests that converge upstream carry identical Reflexive Name Prefixes, they will get aggregated. The forwarder might, just as for any other Interest, decide to either do single or multi-path forwarding of that Interest. If sent multi-path in parallel, these also will reconverge on the reverse path and get aggregated.

For multi-destination, Reflexive Interests might get issued by multiple producers, but they will carry the same Reflexive Name Segment and hence be forwarded using the ingress face of the same Trigger Interest PIT entry until reaching the join point, at which they will get aggregated and thus handled identically to any other Interest(s) subject to aggregation.

6.2. Consumer Implementation Considerations

6.2.1. Data objects returned by the consumer to reflexive name Interests arriving from a producer

The Data objects returned to the producer in response to a Reflexive Interest are normal CCNx/NDN data objects. The object returned in response to a Reflexive Interest is named with its hash as the trailing segment of the Reflexive Interest Name, and hence the scope of the object is under most circumstances meaningful only for the duration of the enclosing Interest-Data interaction. This property is ideal for naming and securing data that is "part of" the enclosing interaction — things like method arguments, authenticators, and key exchange parameters, but not for the creation and naming of objects intended to survive outside the current interaction's scope (c.f. Section 5.3, which describes how to provide globally-named objects using encapsulation). In general, the consumer should use the following guidelines in creating Data messages in response to Reflexive Interest messages from the producer.

(a) Set the recommended cache time (T_CACHETIME) either to zero, or a value no greater than the Interest lifetime (T_INTLIFE) of the Trigger Interest message.

(b) Set the payload type (T_PAYLDTYPE) according to the type of object being returned (e.g. object, link, manifest)

(c) Set the expiry time (T_EXPIRY) to a value greater than *now*, and less than or equal to the *now* + Interest lifetime (T_INTLIFE) of the original Interest message.

6.2.2. Terminating unwanted reflexive Interest exchanges

A consumer may wish to stop receiving Reflexive Interests due to possible errors or malicious behavior on the part of the producer. Therefore, if the consumer receives an unwanted Reflexive Interest, it SHOULD reject that interest with a T_RETURN_PROHIBITED error (See section 10.3.6 of [RFC8609]). This will provoke the forwarders to prevent further reflexive Interests from reaching the consumer, as described above in Section 6.1.2, Paragraph 5.

6.2.3. Interactions with caching

The reflexive named objects provide "local", temporary names that are only defined for one specific interaction between a consumer and a producer. Corresponding Data objects MUST NOT be shared among multiple consumers (violating this would require special gyrations by the producer since the reflexive Name utilizes per-consumer/per-interaction unique values generated randomly as described in Section 4.3). A producer MUST NOT issue an Interest message for any reflexive name after it has sent the final Data message answering the original Interest.

Forwarders MAY still cache reflexive Data objects for retransmissions within a transactions, but they MUST invalidate or remove them from the content store when they forward the final Data message answering the Trigger Interest.

6.3. Producer Implementation Considerations

Producers receiving an Interest with a Reflexive Name Segment, MAY decide to issue Reflexive Interests for the corresponding Data objects. All Reflexive Interest messages that a producer sends MUST be sent over the face that the Trigger Interest was received on.

7. Operational Considerations

Reflexive forwarding represents a substantial enhancement to the CCNx/NDN protocol architecture and hence has important forward and backward compatibility consequences. The most important of these is that correct operation of the scheme requires an unbroken chain of forwarders between the consumer and the desired producer that support Reflexive Name Prefixes and the corresponding forwarder capabilities specified in Section 4.6. When this invariant is not satisfied, some means is necessary to detect and hopefully recover from the error. We have identified three possible approaches to handling the lack of universal deployment of forwarders supporting the reflexive forwarding scheme.

The first approach simply lets the producer detect the error by getting a "no route to destination" error when trying to send an Interest to a reflexive name. This will catch the error, but only after forwarding resources are tied up and the producer has done some work on the Trigger Interest message. Further, the producer would need a bit of smarts to determine that this is a permanent error and not a transient error to be retried. In order for the consumer to attempt recovery, there might be a need for some explicit error returned for the Trigger interest to tell the consumer what the likely problem is. This approach does not enable an obvious recovery path for the consumer either, since if the producer cannot easily detect the error, the consumer has no way to know if a retry has any chance of succeeding.

A second approach is to bump the CCNx/NDN protocol version to explicitly indicate the lack of compatability. Such Interests would be rejected by forwarders not supporting these protocol extensions. A consumer wishing to use Reflexive Name Prefixes would use the higher protocol version on those Interest messages (but could of course continue to use the current version number on other Interest messages). This is a big hammer, but may be called for in this situation because:

(a) it detects the problem immediately and deterministically, and

(b) one could assume an ICN routing protocol that would only forward to a next hop that supports the updated protocol version number. The supported forwarder protocol versions would have been communicated in the routing protocol ahead of time.

A third option is to, as a precondition to utilizing the protocol in a deployment, create and deploy a neighbor capability exchange protocol which will tell a downstream forwarder if the upstream can handle the new protocol elements. This might avoid the large hammer of updating the protocol version, but of course this puts a pretty strong dependency on somebody actually

designing and publishing such a protocol! On the other hand, a neighbor capability exchange protocol for CCNx/NDN would have a number of other substantial benefits, which makes it worth seriously considering anyway.

8. Mapping to CCNx and NDN packet encodings

8.1. Packet encoding for CCNx

For CCNx [RFC8609], this specification defines one new Name Segment type for the Reflexive Name Prefix.

Abbrev	Name	Description
T_REFLEXIVE_NAME	Reflexive Name Segment	Name segment to use as name prefix in Reflexive Interest Messages

Table 1: Reflexive Name Segment

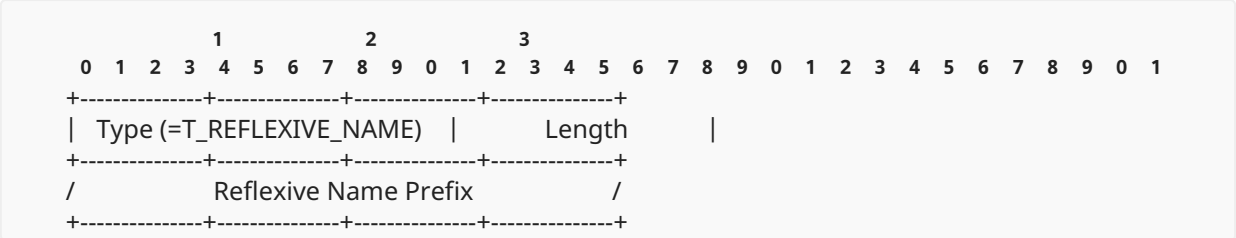


Figure 6: Reflexive Name Prefix

Type: Format of the Value field. For the Reflexive Name Segment, the type value MUST be T_REFLEXIVE_NAME in the current specification. 16 bit length.

Length: Length of Value field in octets. 16 bit length.

Reflexive Name Prefix Reflexive Name Prefix (RNP) assigned by consumer. Variable length.

For Trigger Interests, if approach 2 from Section 4.3.1 is chosen, then a second registered type is needed. The same format as used above for the Name Segment Type will be registered by IANA as an Interest Message TLV.

Abbrev	Name	Description
T_REFLEXIVE_NAME_PREFIX	Reflexive Name Prefix TLV	Interest message field to use as name prefix in Trigger Interest Messages

Table 2: Reflexive Name Prefix TLV

8.2. Packet encoding for NDN

These are proposed assignments based on [NDNTLV]. Suggestions from the NDN team would be greatly appreciated.

8.2.1. Reflexive Name Component Type

The NDN Name component TLVs needs to have a new component type added with type RNP (for reflexive name prefix). We suggest something like: **TBD**

> **Note:**It seems like the current 0.2.1 packet format only has allocated two name component types — a *GenericNameComponent* and a *ImplicitSha256DigestComponent*. Shouldn't there be more types by now or is this spec out of date?

8.2.2. Reflexive Name Prefix TLV

For Trigger Interests, if approach 2 from Section 4.3.1 is chosen, The Reflexive Name Prefix TLV needs to be added to the NDN Interest packet format. We suggest using [RFC9562], hence something like:

RNP ::=	RNP-TYPE	TLV-LENGTH(=16) BYTE8)

Table 3: Proposed Reflexive Name Prefix TLV for NDN Interest Packet

> **[HA]** CCNx adopts variable length for RNP. NDN as well?

9. IANA Considerations

As per [RFC8126], this section makes an assignment in two existing registries in the "Content-Centric Networking (CCNx)" registry group. The registration procedure is "RFC Required", which requires only that this document be published as an RFC.

9.1. CCNx Name Segment Type Registry

As shown in Section 8.1, this document defines Name Segment type, T_REFLEXIVE_NAME, whose suggested value is %x0005.

9.2. CCNx Validation-Dependent Data Types Registry

As shown in Section 8.1, this document defines one CCNx Validation-Dependent Data Type, T_REFLEXIVE_NAME, whose suggested value is %x0008.

10. Security Considerations

One of the major motivations for the reflexive forwarding extension specified in this document is in fact to enable better security and privacy characteristics for ICN networks. The main considerations are presented in Section 1, but we briefly recapitulate them here:

- Current approaches to authentication and data transfer often use payloads in Interest messages, which are clumsy to secure (Interest messages must be signed) and as a consequence make it very difficult to ensure consumer privacy. Reflexive forwarding moves all sensitive data to the Data messages sent in response to reflexive Interests, which are secured in the same manner as all other Data messages in the CCNx and NDN protocol designs.
- In many scenarios, consumers are forced to also act as producers so that data may be fetched by either a particular, or arbitrary other party. Therefore the consumer must arrange to have a routable name prefix and that prefix be disseminated by the routing protocol or other means. This represents both a privacy hazard (by revealing possible important information about the consumer) and a security concern as it opens up the consumer to the full panoply of flooding and crafted Interest Denial of Service attacks.
- In order to achieve multi-way handshakes, in current designs a consumer wishing a producer to communicate back must inform the producer of what (globally routable) name to use. This gives the consumer a convenient means to mount a variety of reflection attacks by recruiting the producer to send Interests to desired victims.

As a major protocol extension however, this design brings its own potential security issues, which are discussed in the following subsections.

10.1. Collisions of reflexive Interest names

Reflexive Interest names need to be constructed with sufficient randomness to ensure an off-path attacker cannot easily manufacture a matching reflexive Interest and either masquerade as the producer, or mount a denial of service attack on the consumer. Doing so also limits tracking through the linkability of Interests containing a re-used random value.

Therefore consumers MUST utilize a robust means of generating these random values, and it is RECOMMENDED that the [RFC9562] format be used, with a pseudo-random number generator (PRNG) approved for use with cryptographic protocols.

10.2. Additional resource pressure on PIT and FIB

Normal Interest message processing in CCNx and NDN needs to consider effect of various resource depletion attacks on the PIT, particularly in the form of Interest flooding attacks (see [Gasti2012] for a good overview of DoS and DDoS mitigation on ICN networks). Interest messages utilizing this reflexive forwarding extension can place additional resource pressure on the PIT.

While this does not represent a new DoS/DDoS attack vector, the ability of a malicious consumer to utilize this extension in an attack does represent an increased risk of resource depletion, especially if such Interests are given unfair access to PIT and FIB resources. Implementers SHOULD therefore protect PIT and FIB resources by weighing requests for reflexive forwarding resources appropriately relative to other Interests.

10.3. Privacy Considerations

ICN architectures like CCNx and NDN provide a rich tapestry of interesting privacy issues, which have been extensively explored in the research literature. The fundamental tradeoffs for privacy concern the risk of exposing the names of information objects to the forwarding elements of the network, which is a necessary property of any name-based routing and forwarding design. Numerous approaches have been explored with varying degrees of success, such as onion routing ([DiBenedettoGTU12]), name encryption ([Ghali2017]), and name obfuscation ([Arianfar2011]) among others.

Reflexive forwarding does not change the overall landscape of privacy tradeoffs, nor seem to introduce additional hazards. In fact, the privacy exposures are confined to the reverse path of forwarders from the producer to the consumer, through which the original Trigger Interest forwarding may have already exposed names on path. Similar name privacy techniques to those cited above may be equally applied to the names in reflexive Interests.

While the individual reflexive Interest-Data exchanges have similar properties to those in any NDN or CCNx exchange, the target usages by applications may have interaction patterns that are subject to relatively straightforward fingerprinting by adversaries. For example, a particular remote method invocation may fingerprint simply through the count of arguments fetched by the producer and their sizes. The attacker must however be on path, which somewhat ameliorates the exposure hazards.

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

12. Informative References

[Arianfar2011] Arianfar, S., Koponen, T., Raghavan, B., and S. Shenker, "On preserving privacy in content-oriented networks", Proc. ACM SIGCOMM workshop on Information-Centric Networking (ICN '11), DOI https://doi.org/10.1145/2018584.2018589, August 2011, <https://dl.acm.org/doi/10.1145/2018584.2018589>.

[Asaeda2019] Asaeda, H., Ooka, A., Matsuzono, K., and R. Li, "Cefore: Software Platform Enabling Content-Centric Networking and Beyond", IEICE Transactions on Communications, Vol.E102-B, No.9, DOI 10.1587/transcom.2018EII0001, September 2019, <https://search.ieice.org/bin/summary.php?id=e102-b_9_1792>.

[Auge2018] Augé, J., Carofiglio, G., Grassi, G., Muscariello, L., Pau, G., and X. Zeng, "MAP-Me: Managing Anchor-Less Producer Mobility in Content-Centric Networks", IEEE Transactions on Network, Volume 15, Issue 2, DOI 10.1109/TNSM.2018.2796720, June 2018, <https://ieeexplore.ieee.org/document/8267132>.

[Baccelli2014] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, T., and M. Wählisch, "Information centric networking in the IoT: experiments with NDN in the wild", Proc. the 1st ACM Conference on Information-Centric Networking (ICN '14), DOI 10.1145/2660129.2660144, September 2014, <https://dl.acm.org/doi/abs/10.1145/2660129.2660144>.

[Carzaniga2011] Carzaniga, A., Papalini, M., and A.L. Wolf, "Content-Based Publish/Subscribe Networking and Information-Centric Networking", DOI 10.1145/2018584.2018599, September 2011, <https://conferences.sigcomm.org/sigcomm/2011/papers/icn/p56.pdf>.

[Cefore] "Cefore", <https://github.com/cefore/>.

[Chen2015] Chen, S., Cao, J., and L. Zhu, "NDSS: A Named Data Storage System, in International Conference on Cloud and Autonomic Computing", DOI 10.1109/ICCAC.2015.12, September 2014, <https://ieeexplore.ieee.org/document/7312154>.

[DiBenedettoGTU12] DiBenedetto, S., Gasti, P., Tsudik, G., and E. Uzun, "ANDaNA: Anonymous Named Data Networking Application, in NDSS 2012", DOI https://arxiv.org/abs/1112.2205v2, 2102, <https://www.ndss-symposium.org/ndss2012/andana-anonymous-named-data-networking-application>.

[Gasti2012] Gasti, P., Tsudik, G., Uzun, Ersin., and L. Zhang, "DoS and DDoS in Named Data Networking", Proc. the 22nd International Conference on Computer Communication and Networks (ICCCN), DOI 10.1109/ICCCN.2013.6614127, August 2013, <https://ieeexplore.ieee.org/document/6614127>.

[Ghali2017] Tsudik, G., Ghali, C., and C. Wood, "When encryption is not enough: privacy attacks in content-centric networking", Proc. the 4th ACM Conference on Information-Centric Networking (ICN '17), DOI https://doi.org/10.1145/3125719.3125723, September 2017, <https://dl.acm.org/doi/abs/10.1145/3125719.3125723>.

[Gundogan2018] Gündoğan, C., Kietzmann, P., Schmidt, T., and M. Wählisch, "HoPP: publish-subscribe for the constrained IoT", Proc. the 5th ACM Conference on Information-Centric Networking (ICN '18), DOI 10.1145/3267955.3269020, September 2018, <https://dl.acm.org/doi/abs/10.1145/3267955.3269020>.

[I-D.irtf-icnrg-ccnxchunking] Mosko, M. and H. Asaeda, "CCNx Content Object Chunking", Work in Progress, Internet-Draft, draft-irtf-icnrg-ccnxchunking-00, 3 March 2025, <https://datatracker.ietf.org/api/v1/doc/document/draft-irtf-icnrg-ccnxchunking/>.

[I-D.irtf-icnrg-flic] Tschudin, C., Wood, C. A., Mosko, M., and D. Oran, "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-07, 3 March 2025, <https://datatracker.ietf.org/api/v1/doc/document/draft-irtf-icnrg-flic/>.

[Krol2018] Krol, M., Habak, K., Oran, D., Kutscher, D., and I. Psaras, "RICE: Remote Method Invocation in ICN", Proc. the 5th ACM Conference on Information-Centric Networking (ICN '18), DOI 10.1145/3267955.3267956, September 2018, <https://conferences.sigcomm.org/acm-icn/2018/proceedings/icn18-final9.pdf>.

[Kutscher2022] Kutscher, D. and D. Oran, "RESTful information-centric networking: statement", Proc. the 9th ACM Conference on Information-Centric Networking (ICN '22), DOI https://doi.org/10.1145/3517212.3558089, September 2022, <https://dl.acm.org/doi/10.1145/3517212.3558089>.

[Lindgren2016] Lindgren, A., Ben Abdessiem, F., Ahlgren, B., Schlelén, O., and A.M. Malik, "Design choices for the IoT in Information-Centric Networks", Proc. the 13th IEEE Annual Consumer Communications and Networking Conference (CCNC 2016), DOI 10.1109/CCNC.2016.7444905, January 2016, <https://ieeexplore.ieee.org/abstract/document/7444905>.

[Moiseenko2014] Moiseenko, I., Stapp, M., and D. Oran, "Communication patterns for web interaction in named data networking", DOI 10.1145/2660129.2660152, September 2014, <https://dl.acm.org/doi/10.1145/2660129.2660152>.

[Mosko2017] Mosko, M., "CCNx 1.0 Bidirectional Streams", arXiv 1707.04738, July 2017, <https://arxiv.org/abs/1707.04738>.

[NDN] "Named Data Networking", 2020, <https://named-data.net/project/execsummary/>.

[NDNTLV] "NDN Packet Format Specification", 2016, <http://named-data.net/doc/ndn-tlv/>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC6337] Okumura, S., Sawada, T., and P. Kyzivat, "Session Initiation Protocol (SIP) Usage of the Offer/Answer Model", RFC 6337, DOI 10.17487/RFC6337, August 2011, <https://www.rfc-editor.org/info/rfc6337>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <https://www.rfc-editor.org/info/rfc7530>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8793] Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): Content-Centric Networking (CCNx) and Named Data Networking (NDN) Terminology", RFC 8793, DOI 10.17487/RFC8793, June 2020, <https://www.rfc-editor.org/info/rfc8793>.

[RFC9531] Moiseenko, I. and D. Oran, "Path Steering in Content-Centric Networking (CCNx) and Named Data Networking (NDN)", RFC 9531, DOI 10.17487/RFC9531, March 2024, <https://www.rfc-editor.org/info/rfc9531>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/info/rfc9562>.

[Shi2020] Shi, J., Pesavento, D., and L. Benmohamed, "NDN-DPDK: NDN Forwarding at 100 Gbps on Commodity Hardware", Proc. the 7th ACM Conference on Information-Centric Networking (ICN '20), DOI 10.1145/3405656.3418715, September 2020, <https://dl.acm.org/doi/10.1145/3405656.3418715>.

[So2013] So, W., Narayanan, A., and D. Oran, "Named data networking on a router: Fast and DoS-resistant forwarding with hash tables, in proceedings of Architectures for Networking and Communications Systems", DOI 10.1109/ANCS.2013.6665203, October 2013, <https://ieeexplore.ieee.org/document/6665203>.

[Zhang2018] Zhang, Y., Xia, Z., Mastorakis, S., and L. Zhang, "KITE: Producer Mobility Support in Named Data Networking", Proc. the 5th ACM Conference on Information-Centric Networking (ICN '18), DOI 10.1145/3267955.3267959, September 2018, <https://conferences.sigcomm.org/acm-icn/2018/proceedings/icn18-final23.pdf>.

Appendix A. Alternative Designs Considered

During development of this specification, a number of alternative designs were considered and at least partially documented. This appendix explains them for historical purposes, and explains why these were considered inferior to the design we settled on to carry forward.

A.1. Handling reflexive interests using dynamic FIB entries

The original draft specification employed the use of dynamically-created FIB entries for forwarding Reflexive Interests. In this approach, at each forwarder along the reverse path from producer to consumer, a FIB entry must be present that matches this name via Longest Name Prefix Match (LNPM), so that when the reflexive interest arrives, the forwarder can forward it downstream toward the originating consumer. This FIB entry would point directly to the incoming interface on which the corresponding original Interest arrived. The FIB entry needs to be created as part of the forwarding of the original Interest so that it is available in time to catch any reflexive Interest issued by the producer. It would usually make sense to destroy this FIB entry when the Data message satisfying the original Interest arrives since this avoids any dangling stale state. Given the design details discussed below, stale FIB state would not represent a correctness hazard and hence could be done lazily if desired in an implementation.

In this scheme, the forwarder operates as follows:

1. The forwarder creates short-lifetime FIB entries for any Reflexive Interest Name prefixes communicated in an Interest message. If the forwarder does not have sufficient resources to do so, it rejects the Interest with the T_RETURN_NO_RESOURCES error — the same error used if the forwarder were lacking sufficient PIT resources to process the Interest message.
2. Those FIB entries are queried whenever an Interest message arrives whose first name segment is of the type *Reflexive Name Segment (RNP)*
3. The FIB entry gets removed eventually, after the corresponding Data message has been forwarded. One option would be to remove the FIB directly after the Data message has been forwarded. However, the forwarder might choose do lazy cleanup.

There are a number of additional considerations with this design that need to be dealt with.

A.1.1. Design complexities and performance issues with FIB-based design

When processing an Interest containing the reflexive name TLV and creating the necessary FIB entry, the forwarder also creates a *back pointer* from that FIB entry to the PIT entry for the Interest message that created it. This PIT entry contains the current value of the remaining Interest lifetime or alternatively a value from which the remaining Interest lifetime can be easily computed. Call this value IL_t.

The forwarder input thread could key off the high-order name segment type (one byte) and if reflexive, do a reflexive FIB lookup instead of a full name hash. The reflexive FIB entry would contain the shard identity of the matching Interest (concretely, the core id servicing the shard) and steer the reflexive interest there. The reflexive name prefix FIB lookup would have to be competitive performance-wise with a full-name hash for this to win, however. Experimentation is needed to further evaluate such implementation tradeoffs for input packet load balancing in high-speed forwarders.

The FIB is a performance-critical data structure in any forwarder, as it needs to support relatively expensive longest name prefix match (LNPM) lookup algorithms. A number of well-known FIB data structures are heavily optimized for read access, since for normal Interest message processing the FIB changes slowly — only after topological changes or routing protocol updates. Support for reflexive names using dynamic FIB entries changes this, as FIB entries would be created and destroyed rapidly as Interest messages containing reflexive name TLVs are processed and the corresponding Data messages come back.

While it may be feasible, especially in low-end forwarders handling a low packet forwarding rate to ignore this problem, for high-speed forwarders there are a number of hazards, including:

1. If the entire FIB needs to be locked in order to insert or remove entries, this could cause inflated forwarding delays or in extreme cases, forwarding performance collapse.
2. A number of high-speed forwarder implementations employ a sharded PIT scheme (see Section 6.1.1, Paragraph 4) to better parallelize forwarding across processing cores. The FIB, however, is still a shared data structure which is either read without read locks across cores, or explicitly copied such that there is a separate copy of the FIB for each PIT shard. Clearly, a high update rate without read locks and/or updating many copies of the FIB are unattractive implementation options. (Note: unlike the adopted scheme in the main specification, by just depending on a dynamic FIB it is not feasible to force reflexive interests to be hashed or be otherwise directed to the PIT shard holding the original Interest state).

There are any number of alternative FIB implementations that can work adequately. The most straightforward would be to simply implement a "special" FIB for just reflexive name lookups. This is feasible because reflexive names deterministically contain the distinguished high-order name segment type of T_REFLEXIVE_NAME, whose content is a 64-bit value that can be easily hashed to a FIB entry directly, avoiding the more expensive LNPM lookup. Inserts and deletes then devolve to the well-understood problem of hash table maintenance.

A.1.2. Interactions between FIB-based design and Interest Lifetime

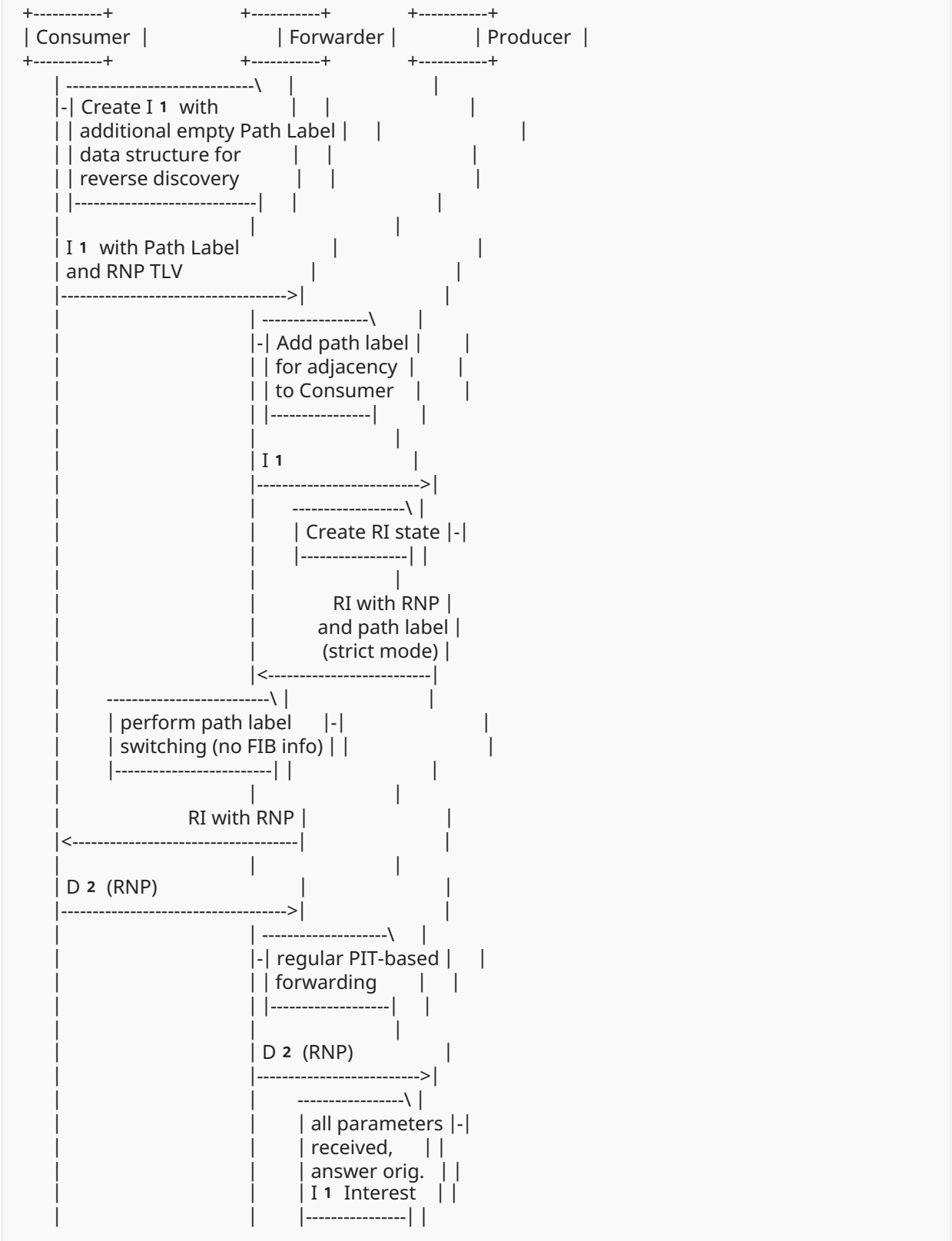
If Interest lifetime handling is implemented naively, it may run afoul of a sharded PIT forwarder implementation, since the PIT entry for the reflexive Interest and the PIT entry for the original Interest may be in different shards. Therefore, if the update is done cross-shard on each reflexive Interest arrival, performance may suffer, perhaps dramatically. Instead, the following approach to updating the Interest lifetime after computing the new value is would be needed by this FIB-based design for sharded-PIT forwarders.

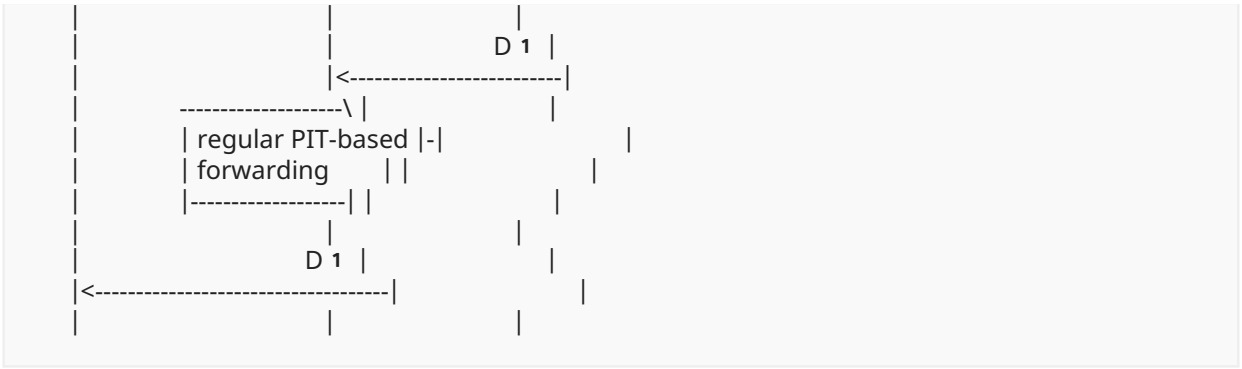
When creating the reflexive FIB entry as above in Appendix A.1.1, copy the remaining Interest lifetime from the PIT entry. Do the PIT update if and only if this value is about to expire, thus paying the cross-shard update cost only if the original Interest is about to expire. A further optimization at the cost of modest extra complexity is to instead *queue* the update to the core holding the shard of the original PIT entry rather than doing the update directly. If the PIT entry expires or is satisfied, instead of removing it the associated core checks the update queue and does the necessary update.

A.2. Reflexive forwarding using Path Steering

We also considered leveraging Path Steering [RFC9531] Path Labels that inform the forwarder at each hop which outgoing face to use for for forwarding the Reflexive Interest. In this approach, the producer, when creating and issuing the Reflexive Interest with the Reflexive Name Prefix includes a Path Label to strictly steer the forwarding at all hops from the producer to the consumer (strict mode Path Steering). This means, the Reflexive Interest carries the Reflexive Name Prefix, but forwarders do not apply LNPM or any other outgoing face selection based on the name. It also eliminates the need for dynamic FIB entries as discussed above in Appendix A.1. Instead the forwarding is strictly steered by the Path Label using regular Path Steering semantics.

The message flow using Path Steering would look like the following:


Consumer
Forwarder
Producer
Create I 1 with additional empty Path Label data structure for reverse discovery
I 1 with Path Label and RNP TLV
Add path label for adjacency to Consumer
I 1
Create RI state
RI with RNP and path label (strict mode)
perform path label switching (no FIB info)
RI with RNP
D 2 (RNP)
regular PIT-based forwarding
D 2 (RNP)
all parameters received, answer orig.
I 1 Interest



Legend:
I 1 : Interest # 1 containing the Reflexive Name Prefix TLV
RI: Reflexive Interest with Reflexive Name Prefix Segment
RNP: Reflexive Name Prefix
D 1 : Data message, answering initiating I 1 Interest
D 2 : Data message, answering RI

Figure 7: Message Flow Overview using Path Steering

Path Steering uses Path Label data structures on the downstream path (from producer to consumer) to discover and collect hop-by-hop forwarding information so that consumers can then specify selected paths for subsequent Interests. Reflexive Forwarding would use the same data structure, but for "reverse discovery", i.e., in the upstream direction from consumer to producer.

From an operational perspective the path-steering approach does not exhibit good properties with respect to backward compatibility. Without a complete path of forwarders between consumer and producer that support path steering, reflexive interests cannot reach the intended consumer. While we might envision a way to steer a subsequent Interest onto a working path as proposed in [RFC9531], there is no capability to force Interest routing away from an otherwise working path not supporting the reflexive name TLV.

A.3. Multiple RNPs in a Trigger Interest

Early versions of the specification has an option to permit multiple Reflexive Name Prefixes in a Trigger Interest message if none of the other 3 options in Section 4.3 covered the desired use case. The current specification does not permit this, because:

- It inflates the Trigger interest size, and would also inflate the trigger Data reponse size if the RNP is encoded as a name segment in the trigger interest name.
- It is more likely a forwarder will reject the Interest for lack of resources.
- No compelling use case has been found, at least so far

Appendix B. NDN Implementation

An implementation of Reflexive Forwarind in NDN is available at https://github.com/nflsjxc/Reflexive-Forwarding-NDN-demo/tree/master. It differs from this specification in the following ways:

B.1. Reflexive Name Segment

Currently Reflexive Name Component is the suffix instead of the prefix for the Reflexive Names. This is easier for prefix matching in NFD.

B.2. Processing Reflexive Interests>

Currently we omit the RNP checking of Reflexive Interest with RNP in PIT entry because NFD now uses name-based PITs.

Generally speaking there are 2 types of interests in the message flow:

1. Reflexive Interest (Consumer -> Producer)
2. Reflexive Interest (Producer -> Consumer)

Currently the two types of Interests are identified by the Reflexive Name value, if RN=9999, then it is the Reflexive Interest from Producer to Consumer.

Authors' Addresses

Dave Oran
Network Systems Research and Design
4 Shady Hill Square
Cambridge, MA 02138
United States of America
Email: daveoran@orandom.net

Dirk Kutscher
HKUST(GZ)
Guangzhou
China
Email: ietf@dkutscher.net
URI: https://dirk-kutscher.info

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Tokyo
184-8795
Japan
Email: asaeda@nict.go.jp

Kenneth Calvert
University of Kentucky
289 Rose Street
Lexington, KY 40506-0495
United States of America
Email: calvert@netlab.uky.edu