Reflexive Forwarding for CCNx and NDN Protocols

https://datatracker.ietf.org/doc/draft oran icnrg reflexive-forwarding/02/

Dave Oran
Dirk Kutscher

ICN RG @ IETF-113
Motivation

- Many scenarios benefit from ICN’s robust and secure two-way exchange through INTEREST/DATA
- There are other scenarios though where that is not sufficient
  - RESTful communication, e.g., Web over ICN
  - Remote Method Invocation
  - Phone-home scenarios
  - Peer state synchronization
- Desirable features
  - Pushing Data
  - RESTful-like session continuation
- Our goal: enable these scenarios in an ICN-idiomatic way
  - As a foundation for the scenarios above and more
  - Most relevant (probably): RESTful ICN
Application Layer Interactions

- **Web**
  - RESTful communications: series of requests in session context – through representational state transfer
  - Considerable request sizes: header fields, cookies, input data (GET, PUT, POST)

- **Remote Method Invocation**
  - Authentication/authorization info
  - Potentially really large input parameters – think “map-reduce”
Motivations for multi-way Handshakes

• Remote Method Invocation (RMI, aka RPC)
  – Fetch arguments
  – Perform authorization
  – Separate invocation from results return

• Phone-home for sensor/actuators
  – Fetch from gateway rather than push from device
  – Eliminate polling

• Peer State Synchronization
  – 3-way (or more) handshakes needed to avoid hazards
  – Complicated state machines for things needing negotiation (e.g. SIP/SDP)
Requests parameters in INTEREST messages?

- Large input data – not advisable
  - Flow balance
  - Computational overload attacks (server has to process arbitrary client data…)
  - Extra state on forwarders
  - Potential INTEREST fragmentation
Reverse INTEREST for Parameters to Consumer?

- Would require consumer identity (disclosure) with routable prefix
  - Not idiomatic in ICN (no source addresses/names)
  - Consumer mobility much harder
  - Potential reflection attacks (consumer can provide arbitrary "paramter prefix")
- Correlating two independent INTEREST/DATA exchange complicates state machine on both sides
  - Catastrophic if done wrong for key exchange
Design Overview

• Utilize forwarder state established by Interest sent from consumer to producer
  – Allow for not just a returning Data message, but a Reflexive Interest to flow from producer to the unique consumer who sent the original Interest

• Define a scheme for Reflexive Name Prefixes
  – Can only be seen and understood by already established consumer/producer pairing
  – Do not reveal consumer identity (temporary names within the RI context)

• Provide forwarder mechanism for routing these back to consumer from producer

• Couple state of the original Interest/Data exchange with the reflexive exchange(s)
  – Ensure state gets mapped correctly by both consumer and producer
  – And unwound properly at forwarders when Data message responding to original Interest is sent back
High-Level Protocol Overview

Consumer

Forwarder

Producer

I1 [P=P1, RNP=X1]

PIT [P=P1]

RI [P=X1]

I1 State [RNP=X1]

DR [P=X1]

PIT [P=X1]

D1
New Approach (version 02)

- **PIT Tokens for reverse forwarding**
  - Much more efficient PIT lookups
  - No special RFIB forwarder requirements
- **Forward Direction PIT tokens (FPTs)**
  - Attached to
    - Forwarded Interests in upstream direction
    - Forwarded Reflexive Interests in downstream direction
- **Reverse Direction PIT tokens (RPTs)**
  - Attached to
    - Reflexive Interests in downstream direction
    - Data responses to Interests in both directions
New Approach (version 02)

I1[p=p1, RNP=X1]

Record RNP in PIT

Add FPT to I1

PIT [p=p1; RNP=X1]

Use RPT(I2) to find PIT of I1, Checking matching of RNP; Create I2 PIT entry; Forward I2

I2[p=x1, RPT=FPT(I1)]

Add RPT(I2) and FPT (I1) to I2

Consuming I2 PIT state

D2[p=x1]

Consuming I1 PIT state

D1

Remove I1 State
Naming of Reflexive Interests

- New Name Component type for CCNx and NDN
  - High-order component of any reflexive name, used to form prefix

- Value is a 128-bit random number
  - Entropy to uniquely identify the consumer for duration of the exchange
  - Different value for each outer exchange limits linkability
    - UUID (RFC4122)

- Possible reflexive names that can be constructed:
  - A single full name of object to fetch
  - Prefix out of which producer/consumer name multiple objects
  - Full name of a FLIC Manifest
New Node Behavior

• Consumer, Producers, Forwarders

• Forwarder modifications include PIT Token generation when receiving INTERESTs with Reflexive name prefix

• All modifications should be doable for high-performance and standard software-based forwarders

• Details in the draft
## CCNx Encoding

### Reflexive Name TLV

<table>
<thead>
<tr>
<th>Abbrev</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_REFLEXIVE_NAME</td>
<td>Reflexive Name Component</td>
<td>Name component to use as name prefix in Reflexive Interest Messages</td>
</tr>
</tbody>
</table>

### Hop-by-hop PIT Token TLVs

<table>
<thead>
<tr>
<th>Abbrev</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_FPT</td>
<td>Forward PIT TOKEN</td>
<td>1-32 byte value chosen by the forwarder for a PIT entry communicated upstream toward a producer</td>
</tr>
<tr>
<td>T_RPT</td>
<td>Reverse PIT TOKEN</td>
<td>1-32 byte value placed in either a Data packet or a Reflexive Interest packet by a producer or forwarder to allow the upstream forwarder to access the PIT entry identified by a received forward PIT Token (FPT)</td>
</tr>
</tbody>
</table>
NDN Encoding

- Reflexive Name Component Type
  - Need a new component type (type RNP)
- Reflexive Name Prefix TLV
  - \( RNP ::= | RNP-TYPE | TLV-LENGTH(=16) \)\( BYTE8 \)
- PIT Tokens for NDNLPv2
  - Need additional type for reverse PIT token

```
<table>
<thead>
<tr>
<th>LpHeaderField</th>
<th>PitToken</th>
</tr>
</thead>
<tbody>
<tr>
<td>PitToken</td>
<td>PIT-TOKEN-TYPE TLV-LENGTH 1*32OCTET&gt;</td>
</tr>
</tbody>
</table>

Current NDNLPv2 PIT Token

```
```
<table>
<thead>
<tr>
<th>LpHeaderField</th>
<th>ReversePitToken</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReversePitToken</td>
<td>PIT-TOKEN-TYPE TLV-LENGTH 1*32OCTET&gt;</td>
</tr>
</tbody>
</table>

Proposed Reverse PIT Token
```
Typical Use Cases

• Remote Method Invocation
• RESTful Web Interactions
• Data Pull from sensors
Remote Method Invocation

(Pioneered by RICE)

• RICE uses (an earlier version of) Reflexive Interests for the following:
  – Retrieve authentication/authorization information from consumer
  – Fetch arguments to method calls

• Completion can be either:
  – Immediate through the returning Data message, or
  – Deferred to a separate exchange to retrieve results by utilizing Thunks.

• Illustrated on following slide
RMI Example

Consumer

I1 to invoke method/RPC
RD1 with argument 1
RD2 with argument 2
Wait awhile...

Producer

Fetch arguments with Reflexive Interests
RI1 to fetch argument 1
RI2 to fetch argument 2
Commit Resources, return Thunk
D1 with Thunk
Perform Computation
D2 with Result

I2 with Thunk name to fetch results
RESTful Web Interactions

• Only place RESTful request via the URI in the initial Interest
• Get all the parameters, including AuthZ with Reflexive Interests
  – Cookies, Accept-foo headers, other HTTP goop
• Return results via regular Data messages
Data Pull from sensors

• Sensor only needs to act as consumer
• Wake up (on timer or event)
• ”Phone Home” to an application gateway or REPO
• This provokes a Reflexive Interest/Data exchange initiated from the gateway
• Data can either be:
  – Packaged/stored by gateway as the authoritative source
  – Named, encapsulated and signed by sensor itself
Phone Home Data Pull Example

**Sensor Consumer**
- Wake up to Phone home

**Gateway Producer**
- Form Reflexive Interest requesting associated Data
- RI1 to fetch sensor Data
- RD1 return requested data
- Optional D1 to complete Handshake
- Store Result as gateway-named data, or Unwrap globally named Data to put in Repo

**I1 Phone Home to gateway as producer**
Operational Considerations

• This is **NOT** backward-compatible
  – Need an unbroken chain of forwarders that support reflexive forwarding or things don’t work right

• Possible ways to overcome this
  – Ignore the problem; let producers get a *no route* error if they try to send a reflexive interest. This is ugly:
    • how does producer figure out why no route
    • How does he tell consumer that original exchange has failed for this reason – may need a new interest return error
  – Bump the CCNx/NDN protocol version on Interests carrying Reflexive Name Prefix TLVs
    • key off this to send back an error from a back-version forwarder
    • Pretty big hammer!
  – Create a capabilities-exchange protocol so forwarders know capabilities of next hops
    • Lots of work, but we probably need such a thing anyway!
Security Considerations

• This scheme is partly motivated by trying to improve both Security and Privacy:
  – Avoids payloads in Interests that then have to be signed, with associated vulnerability to computational attacks on producers
  – Avoids routable names for consumers so they aren’t exposed to various crafted and flooding attacks
  – Avoids sending names crafted by consumers to producers, which can open up reflection attacks
Some things on Security to Consider

• Collisions of Reflexive Name prefixes
  – Avoid by using a crypto-quality PRNG

• Resource pressure on PIT
  – Interests carrying Reflexive Name prefixes are more slightly expensive in both compute and storage

• Privacy
  – Same concerns about leaking information via names as all other cases for CCNx or NDN
  – Use cases may have message exchange and timing patterns that allow easier linkability than independent exchanges
Outlook

• CCNx Key Exchange
• RESTful communication
• Information-Centric Web

• Multi-protocol cookie concept
  • Many protocols utilize “cookie” concept: key exchange, web etc.
  • Idea: minimize number of RTTs (think QUIC 0-RTT)
  • Provide way to integrate a “cookie map” in I1 Interest
That’s about it. Questions & Comments?

Please review and comment on the draft!!!
Upon receiving an Interest containing a RNP TLV:
   - MUST record RNP as element of PIT entry for that Interest

When forwarding an Interest with RNP TLV:
   - MAY generate FPT and append it to the forwarded Interest to be processed by the next hop

If an Interest contains an RPT:
   - MAY use value to access corresponding PIT entry
   - or do a direct lookup based on the Reflexive Interest Name Prefix
Forwarder Operation (2)

1. MUST check that the high-order Name component of Interest is of type RNP
   - IF NOT, simply process the Interest as a normal non-reflexive Interest
   - ELSE treat as Reflexive Interest
     • Create a new PIT entry for the Reflexive Interest
     • Record the FPT (if any, as for other Interests)
     • Look up ingress face from originating Interest's PIT entry and forward the Reflexive Interest on this single face
       – Append RPT from the ingress face information of original Interest's PIT entry, if any
       – Append FPT TLV to Interest if forwarder requires downstream forwarder to supply an RPT in any returning Data packet for this Reflexive interest
TODO: Implementation: Forwarders

• Interest Input – sharded PITs can be tricky
  – Avoid cross-chard updates when handling reflexive interests, or
  – Force reflexive interests into same shard as original interest

• Interest Lifetime – extended by possibly multiple RTTs
  – Could be hard for consumer to guess a good value
    • Likely result is consumers grossly overestimating with bad effects when Interests can experience undetected loss
  – May need to have forwarder account for this by adjusting interest lifetime of original interest when reflexive interests arrive

• Interest Aggregation – actually this all works out without any changes
  – Like other Interest fields, **MUST** create separate PIT entry if Interests carry different reflexive name prefix values.
Implementation: Consumers

• Decide how to name data returned for an arriving reflexive Interest
  – Use a plain Data message if lifetime is just the enclosing exchange
  – Encapsulate a whole Data message with its own fullname if global visibility/lifetime is desired

• Set other fields appropriately for data useful within the enclosing exchange
  – Recommended cache time zero or small
  – Data expiry no longer than Interest lifetime of original interest

• Terminate unwanted reflexive Interest arrivals
  – Send a *Prohibited* Interest Return error
  – Forwarders with then wipe out the corresponding RFIB entry