Reflexive Forwarding for CCNx and NDN Protocols

draft-oran-icnrg-reflexive-forwarding-01

Dave Oran
Network Systems Research & Design
Dirk Kutscher
University of Applied Sciences Emden/Leer

Outline

- Motivations for multi-way interactions in ICN
- Problems with existing approaches.
- Overview of the Reflexive Forwarding design
- Use Cases for reflexive forwarding
- If time available:
  - Implementation implications
  - Operational considerations
  - Security and Privacy considerations

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 handshakes needed to avoid hazards
  - Complicated state machines for things needing negotiation (e.g. SIP/SDP)

Problems with Existing approaches: Pushing Data

- Interest messages get big
  - Might need fragmentation (ugh!)
  - Messes up assumption of small(ish)interests for congestion control
- Need to sign interests for pushed data to be believed
  - Bigger interest still
  - Computational cost on producer to check signature
- Wasted bandwidth if computation started by pushed data winds up abandoned

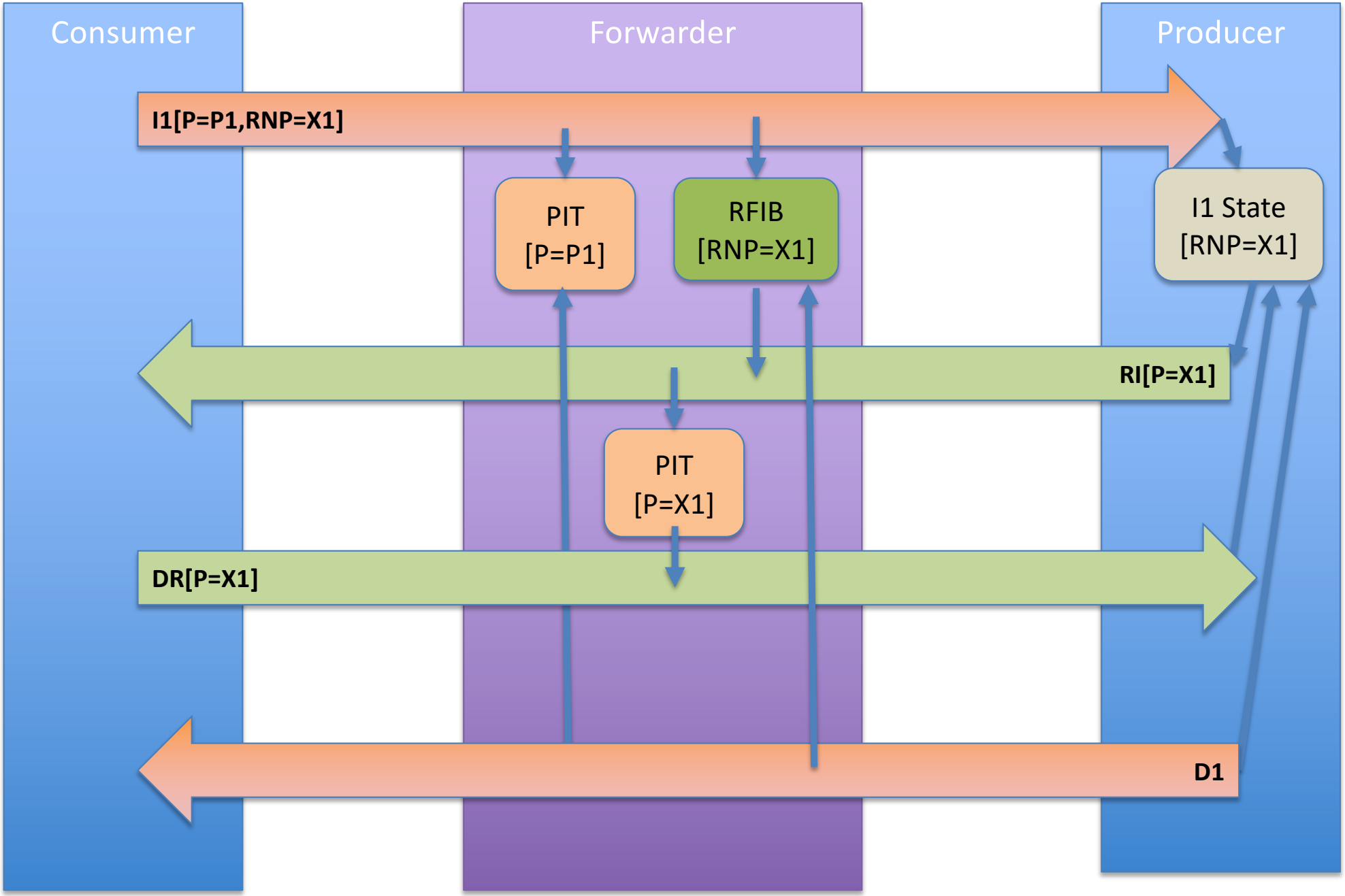
Problems with Existing approaches: Independent Exchanges

- Consumer needs a routable name prefix
  - Exposes consumer to unwanted traffic
  - Puts burden on routing to propagate far enough to reach producer
  - In mobile environments, consumer becomes producer as well, necessitating producer mobility machinery for pure consumer
- Consumer gets to choose the name to use to reach it by
  - Opens up big hole to mount reflection attacks
- Correlating the two independent Interest/Data exchanges can be error-prone
  - Catastrophic if done wrong for key exchange
  - Complicated state machine management (c.f. SIP & SDP)

Design Overview

- Utilize existing chain of PIT breadcrumbs established by an Interest sent from consumer to producer
  - This has enough state to allow 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*
  - These can only be seen and understood by the already established consumer/producer paring
- Provide a FIB enhancement to allow routing these back to the consumer from the producer
- Couple the 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 the forwarders when the Data message responding to the original Interest is sent back

Protocol Walk-through



Naming of Reflexive Interests

- New Name Component type for CCNx
  - High-order component of any reflexive name, used to form prefix
- Value is a 64-bit random number
  - Entropy to uniquely identify the consumer for duration of the exchange
  - Different value for each outer exchange limits linkability
- 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

Forwarder Operation

- Create and manage short-lifetime FIB entries for any reflexive name prefix from an incoming Interest.
- Query these FIB entries (and no others) if an Interest arrives whose first name component is of type Reflexive Name Prefix
- FIB entry consumed along with original PIT entry when the data message is returned by the producer
  - Could be removed lazily due to randomness properties of the values

Typical Use Cases

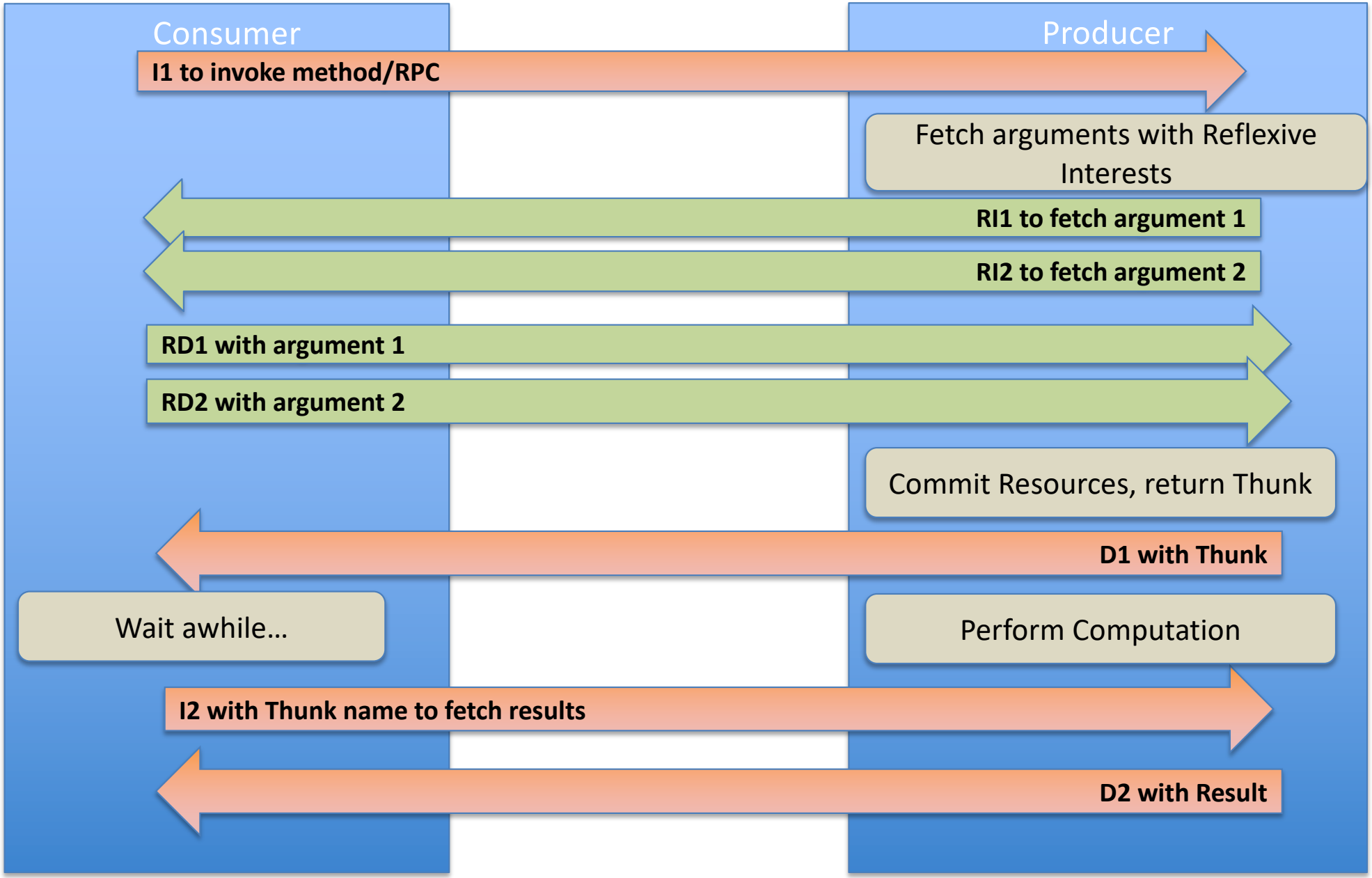
- Remote Method Invocation
- RESTful Web Interactions
- Data Pull from sensors

Remote Method Invocation

(Pioneered by the RICE Remote Invocation on ICN work)

- 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 buy utilizing *Thunks.*
- Illustrated on following slide

RMI Example


Consumer
Producer
I1 to invoke method/RPC
Fetch arguments with Reflexive Interests
RI1 to fetch argument 1
RI2 to fetch argument 2
RD1 with argument 1
RD2 with argument 2
Commit Resources, return Thunk
D1 with Thunk
Wait awhile…
Perform Computation
I2 with Thunk name to fetch results
D2 with Result

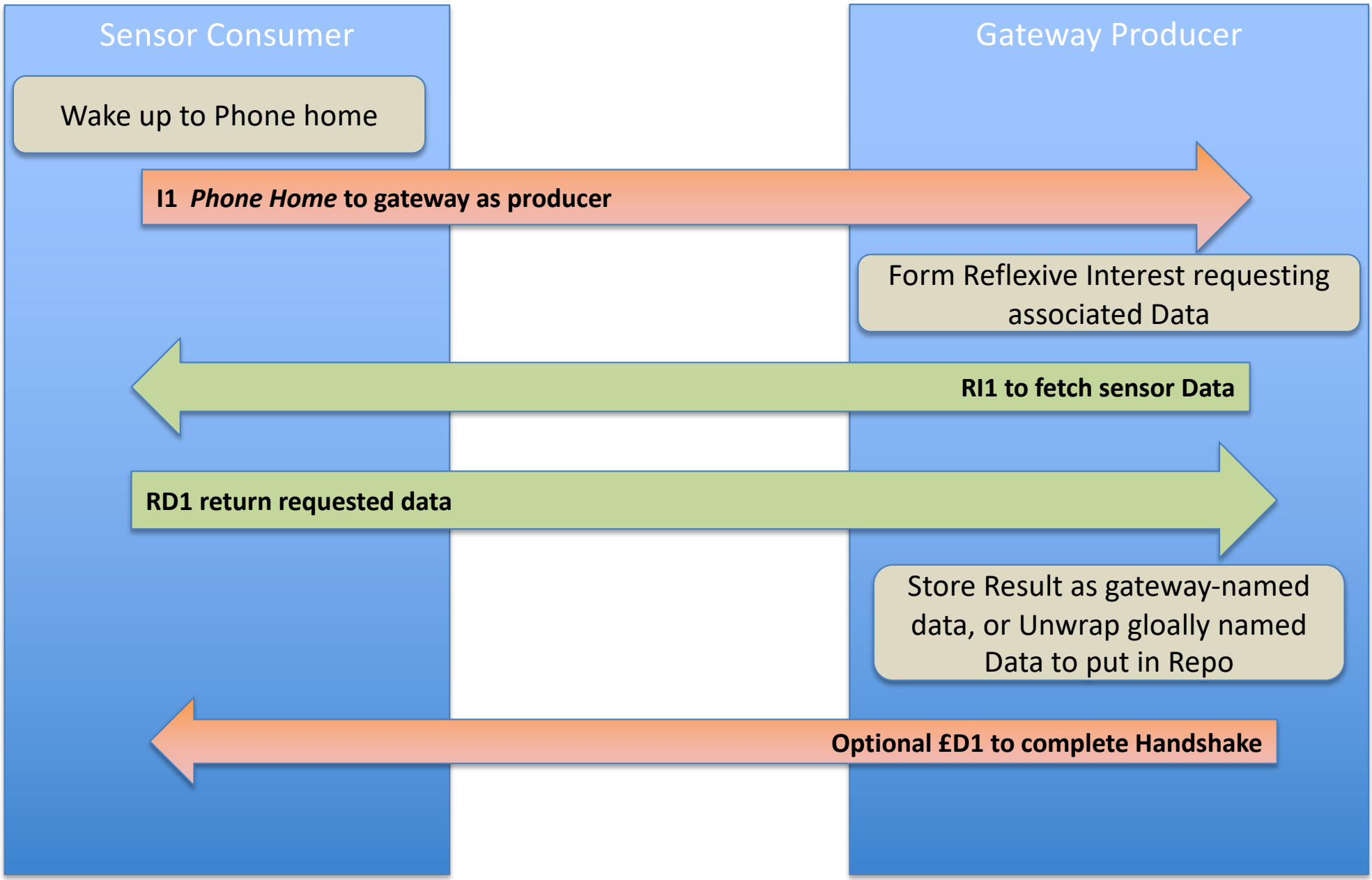
RESTful Web Interactions

- 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



Implementation: Forwarders 1

- FIB - Changes from mostly read, to read-write
  - Probably want a separate data structure – an *RFIB*
  - Not hard because reflexive name component is easily parsed and can be managed with simple 64-bit hashing
- Interest Input – sharded PITs can be tricky
  - Avoid cross-chard updates whebn handling reflexive interests, or
  - Force reflexive interests into same shard as original interest

Implementation: Forwarders 2

- Interest Lifetime – inflated 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
  - Propose to have forwarder account for this by adjusting interest lifetime of original interest when reflexive interests arrive
- Interest Aggregation – surprisingly not a problem
  - Like with other Interest fields, forwarder **MUST** create separate PIT entry if Interests carry different reflexive name prefix values.

Implementation: Consumers

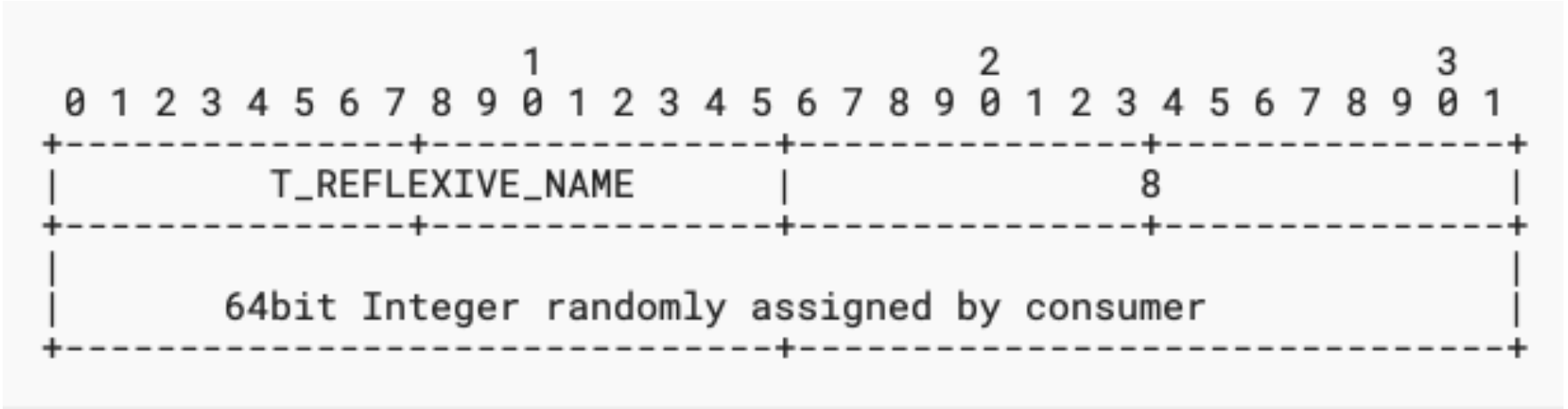
- Decide how to name data returned for an arriving reflexive Interest
  - Use a plain Data message if lifetime is just the enclosing enchange
  - Encapsulate a whole Data message with its own fullname if global visibility/lifetime is desired
- Set other fields appropriately for data useful within the 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

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!

Protocol encoding changes

- This is the simplest part.
  - Just one new Name component type in registry
  - One new Interest TLV to communicate the reflexive name prefix to the forwarders and producer



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 consumersto producers, which can open up reflection atacks

Some things on Security to Consider

- Collisions of Reflexive Name prefixes
  - Avoid by using a crypto-quality PRNG
- Resource pressure on PIT and FIB
  - Interests carrying Reflexive Name prefixes are more expensive in both compute and memory (for the RFIB entry)
- 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

That’s about it.
Questions & Comments?

Please review and comment on the draft!!!