ICN Research Group
Internet-Draft
Intended status: Experimental
Expires: May 7, 2016

A. Lindgren
F. Ben Abdesslem
B. Ahlgren
SICS
O. Schelen
Lulea University of Technology
A. Malik
Ericsson

November 4, 2015

# Proposed Design Choices for IoT over Information Centric Networking
draft-lindgren-icnrg-designchoices-00

## Abstract

This document discusses and describes design choices made in order to utilize Information Centric Networking (ICN) for the Internet of Things (IoT). Based on requirements and challenges identified in draft-zhang-icnrg-iotchallenges-00, we propose design choices for an IoT architecture to handle these requirements, while providing efficiency and scalability. An objective is to, as far as possible, not require IoT specific changes of the ICN architectures per se, but we do indicate some potential modifications of ICN that would improve efficiency and scalability for IoT and other applications.

Furthermore, the document starts outlining how to map the proposed design choices to existing ICN architectures, in a first instance shown for CCN1.0.

## 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 http://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 May 7, 2016.

## Copyright Notice

Table of Contents

## 1. Introduction

Information Centric Networking (ICN) has been shown to efficiently meet current usage demands of computer networks, where users consume content from the network instead of communicating with specific hosts. The applications and usage of the Internet of Things (IoT) often imply information centric usage patterns, where users or devices consume IoT generated content from the network instead of communicating with specific hosts or devices.

However, while the IoT shares many characteristics with typical information centric applications, it differs because of the high heterogeneity of connected devices (including sensors and actuators), the very high rate of new information being generated, and the heterogeneity in requirements from applications regarding information retrieval and dynamic actuation. Because of these differences, using an Information Centric Network to design an architecture of the IoT is often, but not always, beneficial. In practice, the right approach is a complex tradeoff that depends on the applications and usage of the IoT network.

The advantages and inconveniences of using an ICN for the IoT architecture, and the requirements and challenges connected to that are described in [draft-zhang-icnrg-iot-requirements-00]. and helps finding the right tradeoff between using an ICN or an HCN, depending on the context. In this, we explore how to represent and model IoT on top of existing ICN solutions, without requiring IoT specific functionality in the ICN. We discuss this in terms of effectiveness, efficiency and scalability. However, in some cases we do invite for discussion on tentative additions of functionality to ICN in order to make the overall IoT solution more efficient and scalable.

## 2. Proposed Design Choices for IoT over ICN

This section describes some fundamental design choices and trade-offs to allow for effective, efficient and scalable handling of IoT data in an ICN network. An objective with these choices is to facilitate that an ICN network can be used without requiring additions of IoT application specific functionality in the ICN network. However, in some cases we do invite for discussion on tentative additions of functionality to ICN in order to make the overall IoT solution more efficient and scalable.

### 2.1. Relationship to existing Internet protocols

IoT devices can have a role as content generators (e.g., sensors) in where an ICN paradigm should be effective for data retrieval and dissemination. However, IoT devices may also have roles as actuators in which such devices shall be accessed for control purposes. The use of an ICN network may be less natural when actuation and control of specific devices is the key objective. As ICN networks are likely to coexist with existing Internet protocols in most situation, often being deployed as overlay networks, we will consider that there may be situations where a host centric addressing is more suitable for IoT. Thus, to facilitate support of IoT for both data generation and control/actuation, we assume that ICN routing should therefore work in concert with existing Internet protocols. However, we will also investigate the possibility of utilizing ICN network primitives for actuation as well to see what the tradeoffs are, as can be seen in Section 2.12.

### 2.2. Data naming, format and composition

The data served by ICN may be aggregated from smaller components. Although IoT data components in many cases are small and simple, a general challenge in defining ICN applications is to decide how to compose (i.e. group) the data so that it can be effectively named and requested. Requesting partial data inside a composition may become a challenge. Indeed, if data is composed and sub components are requested, which are not directly namable by the requester, finding such a subset will resemble a database query which may require processing to resolve. The ICN network should not have to support such complexity.

A design choice regarding IoT data is therefore to keep the ICN network free from supporting any advanced queries and instead only support directly addressable (i.e., named) data objects. Any advanced composition (hierarchical, graph-based, hyperlink, etc.) of IoT data, and related searching for sub-components, would be handled in servers/endpoints instead of inside the ICN network. The issue of

IoT data structure and searching at that higher level is for further study. For effective ICN interoperability, only the structure of the atomically addressable data objects must be agreed and mapped to the underlying ICN naming scheme. This is to avoid making new requirements on the ICN and to make sure that the need for computation is kept low in the ICN network, essentially limiting it to deciding whether there is a cache hit or not. There are some considerations following from this design choice. First, the size of the directly addressable objects could be kept fairly small to avoid that unwanted data is pulled over resource constrained networks and cached in the ICN network (resulting in better resource utilization, better localization of desired data, and ultimately better scalability). There is however a tradeoff in that smaller data objects results in a larger naming overhead. Second, this approach means that a flat ICN address space would be sufficient, but for practical reasons a hierarchical address space may add some benefits. In any case, there is flexibility in using different addressing schemes depending on what is supported by the existing ICN framework.

## 2.3. Immutable atomic data objects

The number of IoT devices as well as the amount of data produced by these devices may potentially be very large, and the data may be spread over very large ICN networks. The potential problem of cache inconsistencies in an ICN network may therefore be large if we allow for data to be mutable objects. To support scalability and horizontal distribution it is essential to define data properties that facilitate independency and consistency, while minimizing the need for dynamic global synchronization.

A key design choice is therefore to mandate that IoT only uses immutable atomic data objects. This supports large scale distribution by ensuring that there is no stale data in the ICN domain. A cache hit is always a clean hit. A trade-off from this is that dynamic data must be modeled as a stream of immutable data objects, potentially consuming more resources. However, this challenge can be resolved by smart caching strategies where old data is dropped.

There is however some practicalities to consider. Devices, including IoT devices, are restarted now and then. They might in this process loose their state, including what name they used for a particular data value. So in practise it will be hard to implement a strong assumption on immutable data. We therefore likely must be prepared to handle the occasional exception to this rule.

## 2.4. Data naming in streams of immutable data objects

Many IoT devices produce new sensor readings or other data values at regular intervals or on demand. With the design choice on immutable atomic data objects, there is a need to model the resulting stream of sensor readings with a stream of immutable data objects in the ICN domain. The need in this situation is very similar, if not identical, to video streaming, where video frames or chunks are immutable data objects in a video stream.

A key advantage of modelling IoT data as a stream of immutable data objects is that ICN caches will not contain any stale data w r t a given name. However, since new data objects (with new names) representing different versions of a sensor reading may be emitted frequently, there must be a way to tell the different versions apart.

To support immutable streamed data efficiently, while adhering to the expected naming schemes of ICN, we recommend that names of data objects include a sequence number. When data can be named with sequence number, any request may or may not include such a sequence number. If no number is included in the request, the nearest cache hit will result in a response. If a sequence number is included in the request, only an exact cache match will result in a response. A client that wants the "latest" reading can according to our previously mentioned design choice, in Section 2.2, not ask the ICN network such a high level query, instead it must ask for the specific (version of) information. To avoid complicated searching in the ICN nodes, there is thus no way to explicitly ask the network for the "latest" reading, or any other "range" of sequence numbers.

Should a client want the latest reading from a sensor, one method for this is to make a subscription for the pushed stream of data, as described in Section 2.7, provided that the particular ICN architecture supports this interaction model. The confirmation of that subscription can contain the latest reading, and then obviously the normal stream will be received. The reason for including the latest reading in the response is to immediately provide the "state" of sensors that generate new data infrequently.

Another method to obtain the latest reading, or a particular reading in the past, from a sensor is to perform adaptive probing, for example by binary interval reduction. If a requested sequence number does not (yet) exist, there will be a negative answer from the ICN. This method is preferably combined with application knowledge, for example, in the form of capability advertisements as described in Section 2.8 that enable the client to better predict the sequence number to request. The client that always wants the latest value could also dynamically tune its requests for the next data value to

the frequency of the publisher in order to minimise the latency. The fact that non-existing data is asked for would however potentially pose an overload threat to the ICN since each request of non-existing data could result in cache misses that ripple through all the way to the source, which has to respond that the data doesn't exist. It may therefore be beneficial with negative caching. Serving requests for non-existing data is however a generic challenge to ICN (not specifically to IoT) to be resolved.

There is a third method "in between" the above two. If requests for a not yet existing data object can be held for a short time until the data object is actually available, instead of immediately returning "not found", these so called pending requests act as one-time subscriptions. If the ICN supports pending requests (i.e., long lived requests) the consumer may send requests for data that will soon be published (provided that the name of that data can be deduced beforehand). Provided that request aggregation is being used, this mechanism would be efficient and latency-minimising, and at the same time would not require persistent subscription state. It only requires short-lived pending state.

The support for sequence numbers depends on the particular flavor of ICN. The naming scheme of CCN/NDN may here provide an advantage. It is for further study whether it is possible to use ICNs that do not support sequence numbers as part of naming (e.g., by clever use of metadata, namespace, and search functionality) and what the trade-offs would be.

Two issues for further study are the size of the sequence number space and gaps in the sequence numbers. Must sequence number wraparound be handled, or is it possible to require a large enough sequence number space? Wraparound means an exception to the assumption on immutable objects. Gaps in the sequence number space might result in inefficiencies in some of the above methods, or, if the gaps are large, making them unfeasible. Yet, it might not always be possible to guarantee that there are no gaps.

## 2.5. The importance of time

Time is almost always a very important property of IoT data, and especially so for data that change over time. When modeling dynamic IoT data with a stream of immutable data values, it is often the case that a certain IoT data value is a sensor reading at a particular point in time, and the next value in the stream is the next reading in time. Thus, dynamic data is in this case dynamic over time, with well defined (immutable) values for particular points in time.

We therefore argue that it is important to find a way to represent

these time-related streams of immutable data values in ICN. It should be possible to request a data value from a certain time, and to infer/find the name (sequence number) of the latest, most current, data value. The question is whether or not the stream sequence numbers are sufficient to support time. If not, the ICN system needs to be extended with explicit support for time, something we want to avoid. In general, the methods outlined in the previous section are applicable for finding an IoT data value from a particular point in time, including the latest. What is missing is the mapping between sequence number and time.

One possibility could be to use sequence numbers that directly correspond to time, for instance, the Unix (POSIX) time in form of seconds since January 1st, 1970. This would however both limit the time resolution to seconds, and also result in large gaps in the sequence numbers, something that can be problematic, as discussed in the previous section.

There are several other methods for finding readings from a certain time, or the latest reading, for example through a high level request from a server/endpoint, or by using a naming scheme where the name can be directly inferred, e.g., if an IoT device has advertised under which conditions it produces data and how it is named.

To represent absolute time so that it can be directly inferred, one method is that the producer of data in its capability advertisements (Section 2.8) provide a mapping function between sequence number and time. Thereby also readings on the time axis are immutable while it is still possible to efficiently find the latest reading, as described in Section 2.4. It should be noted that sequence numbers then may have gaps in order to cater for triggered non periodic data, etc. Another method is to include meta data with information on absolute time. Using this mapping scheme data from the current second can be efficiently requested (provided that clock synchronisation is accurate enough, which is out of scope of this document).

We also note that time is also important for other applications, in particular for live streaming video. Live video also produces a time-related stream of immutable objects, and would in the same way benefit from such support in the ICN service.

## 2.6. Decoupling and roles of publishers and consumers

Since ICN networks essentially support a request/response model of interaction, we denote the consumers of information as requesters, and the publishers of information as responders. The ICN network in itself provides decoupling of requesters and responders. It is an

important feature of the ICN that it will allow responders (e.g., IoT devices) to be occasionally unreachable (e.g., due to intermittent connectivity, low battery level, duty cycling). Another advantage is that caching in the ICN will ensure that data objects are normally delivered only once from the IoT devices, independently of the number of immediate requesters.

Note however, that the ICN does not (and should not) provide any transformation or aggregation of data. The IoT dissemination architecture should therefore allow for any number of intermediate processing nodes. An intermediate node will be an endpoint in the ICN network that can act as both requester and responder. Such a node may perform aggregation, filtering, selection, etc. The instantiation of such nodes may for example form a directed (acyclic) graph between ultimate responders (IoT devices) and ultimate requesters (the final applications). It is for further study how to define such an architecture.

It is a design choice to keep the IoT dissemination and aggregation functionality outside of the ICN domain. That architecture would be an overlay that may have intricate structure, and put the ICN usage in a new context, where content from ultimate requesters to ultimate responders may go through many IoT processing nodes that collect, process and re-publish data through an ICN for various purposes.

## 2.7. Combination of PULL/PUSH model

A critical decision regarding IoT data is whether to use a PULL model, a PUSH model, or both. In this document, we define a PULL model as a system where data is only sent when explicitly requested, while a PUSH model indicate that data transmission is initiated by the source based on some trigger (either periodic, for each new object, or based on some condition on the generated data). There are some intrinsic trade offs between these models. The PULL model is for example resource efficient when there is an abundant amount of IoT information, potentially redundant from many devices, and the clients only occasionally or partially are interested in the information. The PUSH model is for example efficient when there is real-time information and the clients are interested in all information from specific devices all the time.

A design decision in the IoT domain is to support both PULL and PUSH. The base model should be PULL, since this is the native mode of ICN, meaning that requesters must always start by sending a request. If the request is for some specific data, it can be resolved by returning the data (if it exists). The pull model can be supported efficiently and scalably by an ICN network.

A challenge with the pull model is that it may be inefficient for retrieving new data that occur sporadically or based on specific conditions. Our proposal for an IoT framework is therefore that there must be support for efficiently retrieving such triggered information, without having to poll for it through the ICN. Our proposal is that a request can also include triggers, which means that data will be returned (pushed) when triggers are fulfilled, which may be immediately, or in the future at one or several occasions. This can be used to select alarm conditions, to request continuous or periodic push, etc. The trigger conditions could be set by the requester, or be pre-defined by the responder. The former would be more flexible but also have performance/scalability issues since the number of trigger conditions and consequent data generation would depend on a potential large number of requesters. The latter is more scalable since there will be a predefined and finite number of trigger conditions (as defined in capability advertisements). Our recommended choice, at least for the initial phase, is to go for a simple and scalable solution and therefore adopt the model where available trigger conditions are defined and advertised by the responder. The ICN would be apt for supporting such capability advertisements, given that they are fairly static.

With this, there is no requirement raised on ICNs supporting data push, but we recommend to have a discussion on whether an ICN network can or should provide an option to effectively support a push model of data. Such support could make real-time IoT data dissemination more efficient and scalable as previously mentioned in Section 2.3. However, since we assume that the ICN works with existing IP protocols, such functionality can be provided without ICN, by using traditional unicast or multicast communication. We finally note that an ICN supported push service model would make the ICN network more like a publish/subscribe system.

## 2.8. Capability advertisements

Capability advertisements and discovery can be used by requesters to discover which data is available and/or to which responders to connect to. In a deployment with large numbers of responders, the functionality of automatic advertisement and discovery becomes a critical factor to support scaling. Responders should advertise their methods (inputs, outputs, parameters, triggers, etc) and provide relevant metadata in the responses as advertised. Such capability advertisements should be conservative with resources, which suggests that new advertisements should be posted with reasonably low frequency. This implies that an ICN network can be used for providing capability advertisements. The advertisements should be provided as a stream of immutable objects, or alternatively the IoT system should be tolerant to stale caches. Should there be a

need real-time awareness of dynamic changes, a subscription/push model of capability advertisements could be used as earlier described in Section 2.7.

## 2.9. Name-based routing vs name resolution + 1-step vs 2-step

As described in Section 2.2, the IoT framework should be defined so that new functionality in the ICN is not needed. For data that is frequently generated and regenerated, it makes sense to keep simple structures and provide directly inferable naming/addressing of data objects, so that requesters can directly address the data. For more complex data, such as pre-processed, aggregated and structured data a two-step resolution model is recommended. The IoT devices can provide a higher level resolution based on for example queries and searching, resulting in a number of concrete directly addressable ICN objects. This is similar to what web servers do when they return URLs that requesters can use, but in this case it is named content that is returned.

Consequently, the IoT framework should have no requirement that the ICN network itself should support 2-step addressing (although such 2-step methods may exist in some ICNs)

## 2.10. What’s naming and what’s searching

As described in Section 2.2, the IoT framework should be defined so that no new functionality is required in the ICN for searching data or subcomponents of data. The ICN network supports just naming of atomic data objects, while any searching is provided by the IoT framework, which in itself may be constituted by a highly distributed set of nodes that provide processing, analysis and aggregation of IoT data.

## 2.11. Meta data, tagging/tracing of data

IoT data may be tagged with metadata to tell where it originates from. Tagging is made at the level above the ICN network and may for example be a list of strings. It can be added/changed by the originating node (or a node that assigns the originating ID), and added/changed/deleted by any node that processes the data. The tag can in some cases be used to trace data back to origins. In some cases it makes no sense to request or transmit metadata. For efficiency reasons the ICN network should have support for optional delivery of metadata. This is to be conservative with scarce resources, for example when a wireless node requests data which is cached in the ICN network, it would be beneficial if the requester could tell that it is desirable to not receive any metadata. There should be a discussion whether there should be just one, or more than

one, piece of optional information in ICN content to be future proof.

## 2.12. Handling actuators in the ICN model

If actuators should be controlled using the ICN communication model, we need to map the functionality of the actuator to named data and/or the requesting of named data. We see two main models with some variants as described in the following paragraphs.

In the first model, the state of the actuator is represented by a stream of immutable named data objects. The actuator periodically requests its new state using the name of its designated state object. There then has to be a producer of that state data that responds with the current state. When the actuator receives the response, it sets that new state, invoking its actuation function. Authentication of the producer of the state is important, but as this corresponds directly to publisher and data object authenticity that are fundamental in the ICN model, there are no additional requirements for the IoT domain.

A variant of this first model is that a requester first requests the state of the actuator. The requester supplies additional information with the request including the name of the new state data it will produce. The actuator responds with its state, and then requests its new state using the name that was supplied with the additional information in the first request. This variant enables low latency without high frequency polling.

In the second model, the actuator invokes its actuation function as a side-effect of receiving a particular request. There are several plausible variants. The new state could be encoded in the name of the requested data in the request, or could be supplied as additional information with the request. Regardless, the actuator acts on the new state information as a side effect, and responds with data, possibly its state, to the requester. The security issues are potentially more difficult with this model, since in the ICN model, anyone could make the request. Access control and/or requester authentication are therefore required.

To reap the advantages of caching, it should be possible to cache the state of the actuator in both the aforementioned models. However, we think that caching is not as relevant for actuation as it is for other IoT use cases, and can furthermore even be quite problematic. The first model less so, since the actuator can make sure that its state is arbitrarily fresh with the polling method described in Section 2.4. In other words, the latency until actuation happens can be bounded. The variant of the first model and the second model have larger issues. With caching, it is hard for a requester to make sure

that its request actually reaches the actuator, and thus, it is hard to bound the actuation latency. Some caching directive might be needed in this case for reliable functionality.

## 2.13. Role of constrained IoT devices as ICN nodes

Typical ICN nodes such as routers and gateways are deemed to be rich in resources like energy, processing, bandwidth and storage. IoT devices, on the other hand, are quite constrained in such resources. It is also worth noticing that some resources are more crucial than others. In most cases energy, processing and bandwidth are quite expensive for constrained IoT devices. In contrast, storage has shown a considerably rapid decreasing trend in prices over the past few years. There is reason to believe that the memory needed for IoT devices to act as servers of their data will not be prohibitive and that the data centric role of the devices may be elevated by information-centric networks.

However, it is questionable whether IoT devices also should provide caching for data produced by other IoT devices. In ad-hoc networks this may be desirable, but often there is a desire for wireless nodes to minimize communication by handling only data of their own concern. Our design decision in this regard is that we logically separate IoT server functionality (such as sensing and transmitting IoT data) and ICN functionality (such as routing and caching data generated by other devices). A resource constrained device may choose to only implement IoT functionality and act as a server to the ICN, i.e., not act as intermediate ICN node. However, since storage is getting cheaper, IoT devices should be able to cache their own content and, in essence, act as sources to ICN.

## 3. Security Considerations

The ICN paradigm is information-centric as opposed to state-of-the-art host-centric internet. Besides aspects like naming, content retrieval and caching this also has security implications. ICN advocates the model of trust in content rather than trust in network hosts. This brings in the concept of Object Security which is contrary to session-based security mechanisms such as TLS/DTLS prevalent in the current host-centric internet.

The following sub-sections discuss some security advantages and related design choices of using ICN and Object Security in IoT applications.

### 3.1. Retrieving trusted content from untrusted caches

When functioning in an ICN network, an IoT client is expected to rely on the network to deliver the requested content in an optimal fashion without concerning itself with where the content actually lies. This could potentially mean that each individual object within a stream of immutable objects is retrieved from a different source. Having a trust relationship with each of these different sources is not realistic. This gives rise to the need of retrieving trusted content from untrusted nodes/caches in an ICN network. Through Name-Data Integrity, ICN automatically guarantees data integrity to the requester regardless of the source from where it is delivered. Additionally, Object-based signatures and encryption are ideal in such use cases because it relieves an IoT client application from the hassle of having to establish trust with each node that can potentially cache an IoT object. This also means that a requesting client can make use of more caches in the network, hence resulting in better throughput and latency.

### 3.2. Enabling application-layer processing in untrusted intermediaries

Securing content at the object level provides greater granularity and hence more control. An ICN data object may comprise of several distinct application-layer objects e.g. XML, JSON objects. An example of this is an ICN object that corresponds to all the sensor readings in a certain time interval where each sensor reading is a JSON object. Using Object-based encryption to provide data confidentiality allows for the possibility to encrypt a subset of these application-layer objects while leaving others unencrypted and available for processing in untrusted intermediary nodes (e.g. proxies and caches). With this approach, the IoT application has more control of the parts of data it wants to make public and the parts of data it wants to keep confidential and visible only to peers with the right cryptographic keys.

### 3.3. Energy efficiency of cryptographic mechanisms

Session-based security protocols rely on the exchange of several messages before a secure session is established between a pair of nodes. Use of such protocols in constrained IoT devices can have serious consequences in terms of power efficiency because in most cases transmission and reception of messages is more costly than the cryptographic operations. This is especially true for wireless devices.

The problem is amplified proportionally with the number of nodes the constrained device has to interact with because a secure session would have to be established with every node. If the constrained device is acting as a consumer of data this would mean setting up secure sessions with every caching node that the device retrieves data from. When acting as a producer of data the constrained device would have to setup secure sessions with all the consumers. The Object Security model eliminates this problem because the content is readily available in a secure state in the network. IoT devices producing data can secure it w.r.t. all the intended consumers and start transmitting it right away.

## 4. Mapping Design Choices to CCN

In this section we will describe how to map the design choices proposed in Section 2 to the CCN1.0 architecture.

Work in progress. More content will appear in next version of draft.

## 5. Mapping IoT design requirements to CCN1.0

CCN1.0 enforces exact matching of interest names to content object names. This means that each content object should be asked for with an exact name match and cannot be requested as a wild card. This prevents cache inconsistency. As discussed, IoT sensor data should be modelled as an immutable stream of content objects, published with increasing sequence numbers, with each content object being requested with an exact matching name including the sequence number.

When a sensor node is publishing a stream of content objects with increasing sequence numbers, the latest value can be requested for in CCN1.0 either by adaptive probing or by propagating the interest all the way to the source. The name in the interest message is appended with the hash of the interest payload making it unique, allowing it to propagate to the source without being aggregated. The interest payload can carry publisher specific data either requesting for the latest version of the specified stream or requesting for data generated on-demand. Interest payload could also be used to send commands to actuators. The expiry time of a content object in a stream should be set to match the rate at which the dynamic data is being generated. This prevents older sequence numbered objects from overloading the cache since IoT devices are expected to have small (if any) cache memory.

CCN1.0 does not have native support for push data. Using interests with long lifetimes, one time subscriptions can be made for each content object. This however is not scalable since it leads to a perpetual PIT entry for each requested object in the relays along the path. Since CCN1.0 allows interest payload, push data could be sent as notifications embedded in interest messages. These notifications can be responded to with a content object acknowledgement. This however requires maintaining a multicast path from the source to the consumers.

## 6. Acknowledgements

The work behind and the writing of this document are in part supported by the activity ‘14010 Efficient IoT Content’ within EIT Digital (formerly EIT ICT labs) and the Vinnova GreenIoT project.

## 7. Informative References

[draft-zhang-icnrg-iot-requirements-00]
Zhang, Y., Raychadhuri, D., Grieco, L., Baccelli, E., Burke, J., Ravindran , R., Wang, G., Lindgren, A., Ahlgren, B., and O. Schelen, "Requirements and Challenges for IoT over ICN", Internet Draft draft-zhang-icnrg-iot-requirements-00, November 2015.

Authors' Addresses

Anders F. Lindgren
SICS Swedish ICT
Box 1263
Kista SE-164 29
SE

Phone: +46707177269
Email: andersl@sics.se
URI: http://www.sics.se/~andersl

Fehmi Ben Abdesslem
SICS Swedish ICT
Box 1263
Kista SE-164 29
SE

Phone: +46705470642
Email: fehmi@sics.se
URI: http://www.sics.se/~fehmi

Bengt Ahlgren
SICS Swedish ICT
Box 1263
Kista SE-164 29
SE

Phone: +46703141562
Email: bengta@sics.se
URI: http://www.sics.se/people/bengt-ahlgren

Olov Schelen
Lulea University of Technology
Lulea SE-971 87
SE

Phone:
Email: olov.schelen@ltu.se
URI:

Adeel Mohammad Malik
Ericsson
Kista SE-164 80
SE

Phone: +46725074492
Email: adeel.mohammad.malik@ericsson.com
URI:

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: May 6, 2016

R. Ravindran
A. Chakraborti
Huawei Technologies

November 3, 2015

# Forwarding-Label support in CCN Protocol
## draft-ravi-ccn-forwarding-label-01

Abstract

The objective of this proposal is to enable ID/Locator split in CCN protocol. We enable this through the notion of forwarding-label (FL) object, which is an optional hop-by-hop payload in the Interest message with locator name which identifies a network domain, router, or a host. Depending on the application and trust context FL object is subjected to policy based actions by the forwarders such as invoking security verification or enabling other service-centric actions such as FL object replacement. FL can be inserted by the applications or by the network. To enable dynamic name resolution FL can be modified in the network by designated points such as the edge routers. Enabling ID/Locator split in CCN has several applications such as towards routing optimization, mobility, handling indirections in manifests, and routing scalability.

Table of Contents



## 1. ID/Locator Split in CCN

We discuss here the motivations behind the need for separation between persistent name (ID) and a locator (LID) in the Interest message in the context of CCN and a proposal to achieve this. The advantages of ID/Locator has been extensively studied and has been part of many host-centric protocols such as HIP[2], ILNP [3], and LISP [4] and also is part of FIA architectures such as MobilityFirst[13]. Specifically in CCN, ID based routing is not efficient considering dynamic replication of content, mobile entities, or address the problem of routing scalability [9] issue, hence providing this distinct separation in the protocol offers the following advantages:

- ID and Locator namespaces are managed by different entities. IDs are managed by applications, hence relevant only to consumers ,

producers and intermediate service points, while locator names are managed by network administrator. Locators map to network domains or specific network element through which the named entity is reacheable. The relationship between the two is established during the publishing phase, and managed by a separate name resolution function. ID/Locator distinction in CCN allows applications to manage its own name space and not be restricted by network naming rules.

- Today, CCN Applications bind to persistent IDs, while its resolution is handled by per-hop name-based routing in CCN forwarder using unicast/anycast/broadcast means, with routing scalability linked to name aggregation. This model has issue when the named entity is mobile, migrated, or replicated, as the names have to be announced in the routing control plane introducing instability and churn. Enabling ID/Locator split and managing this mapping in a separate name resolution system shall address the routing churn introduced by dynamic entities. CCN is unique in the sense that both name-based routing and resolution system can operate simultaneously driven by its use based on a given context, for e.g. while inter-domain routing can be handled using name resolution system, intra-domain routing can be based on name-based routing.

- Affording ID/Locator split in an Interest message offers many advantages in many network and application functions such as towards name resolution optimization, mobility, handling indirections in manifests [6], and routing scalability.

Considering the above requirements, we propose Forwarding-label (FL) object which provides flexibility to forward Interests on a name other than the one in the Interest message with the ability to modify it in the network. Handling ID/Locator mapping requires a control plane infrastructure and appropriate network layer state with security functions to avoid malicious usage. Specific control plane or security mechanism of ID/Locator mapping is out of the scope of this document as many techniques can be used towards achieving this. This draft presents various considerations towards FL management (insertion/modification/deletion), processing by a CCN forwarder, PIT/CS implications, FL packet format, and security/trust and discusses its application in various scenarios.

## 2. Forwarding-Label Management

FL is used in scenarios where routing by Interest name for name resolution when dynamic scenarios are considered which include replicated content, device mobility, or where scalability challenges exist when ID based routing is employed. FL objects are subjected to

processing and modification in the network depending specific use case scenario. Following we discuss various aspects of FL related to its semantics and management.

## 2.1. FL Naming

FL are container objects which include LID, service specific metadata, and security attributes for authentication. LIDs are hierarchically structured topologically names where the names follow the definition in [1]. The security attributes are optional and may include validation payload and algorithm as discussed in [1].

## 2.2. FL Insertion

A FL object can be inserted in an Interest message by the consuming application or by the network.

In general in CCN, applications requests a service or content by ID, which is feasible today due to the aggregatable nature of ID. But in certain situations, the application logic may use a FL object in addition to the ID in the Interest message or this action may also be triggered because of feedback from the network on failing to route the Interest message based on ID . Networks which processes traffic from applications outside its trust domain require a way to validate the FL object, one of which is discussed in [9]. Another possibility is that networks may ignore FL object from untrusted applications and only choose to route by the Interest ID.

This FL object insertion can also be triggered at the ingress points of a network domain. Network inserts a FL to an incoming Interest message if the Interest message satisfies flow service profiles imposed by the network administrator in the ingress routers, these services functions include mobility or special handling for content distribution. These service profile matching actions include matching Interest name to service prefixes or triggered by certain marking in the Interest message. FL object inserted within trust domain require may not require security validation.

In situations where a forwarder handles both these scenarios, policies can be applied in the ingress router to handle the two cases appropriately. These policies include the face on which the Interests arrives on, Interest ID etc.

## 2.3. FL Swapping

One FL object can be swapped by another in the network in the context of a given service by designated points in the network. As FL objects carry a LID, and with appropriate representation and security

considerations in the Interest message, FL objects also can be potentially stacked if the Interest message has to be tunneled through a domain where routing based on the current FL object is not applicable.

### 2.4. FL Termination

FL objects are terminated by a forwarder when the LID in it matches its own set of names, here we assume a forwarder could have many LIDs such as domain-ID or router-ID. For e.g. a forwarder in a domain identified as /att/santaclara can process FL object with LID set to this domain name or forwarder ID such as /att/santaclara/pop-x. Whenever a FL object is terminated by the forwarder, depending on the service context, it can attach a new FL object, or conduct processing based on the Interest ID.

## 3. FL Message Format

As FL object is swappable in the network, hence it is proposed as a hop-by-hop field in the optional body of the fixed header as shown in Figure 1. The optional FL container includes attribute of type T_LID_NAME, where the LID (Figure 2) are hierarchically structured variable length ID [1]. A LID implies an locator such as an AS-, Gateway-, Router- or Host- ID. In addition to the LID, optional FL metadata includes information on the application or service context to aid network to invoke appropriate FL processing, such as trust validation of the FL object. Optional security attributes such as authentication information can be included depending on specific use case scenarios, such as secure name delegation information discussed in [9], or signature of the consumer.

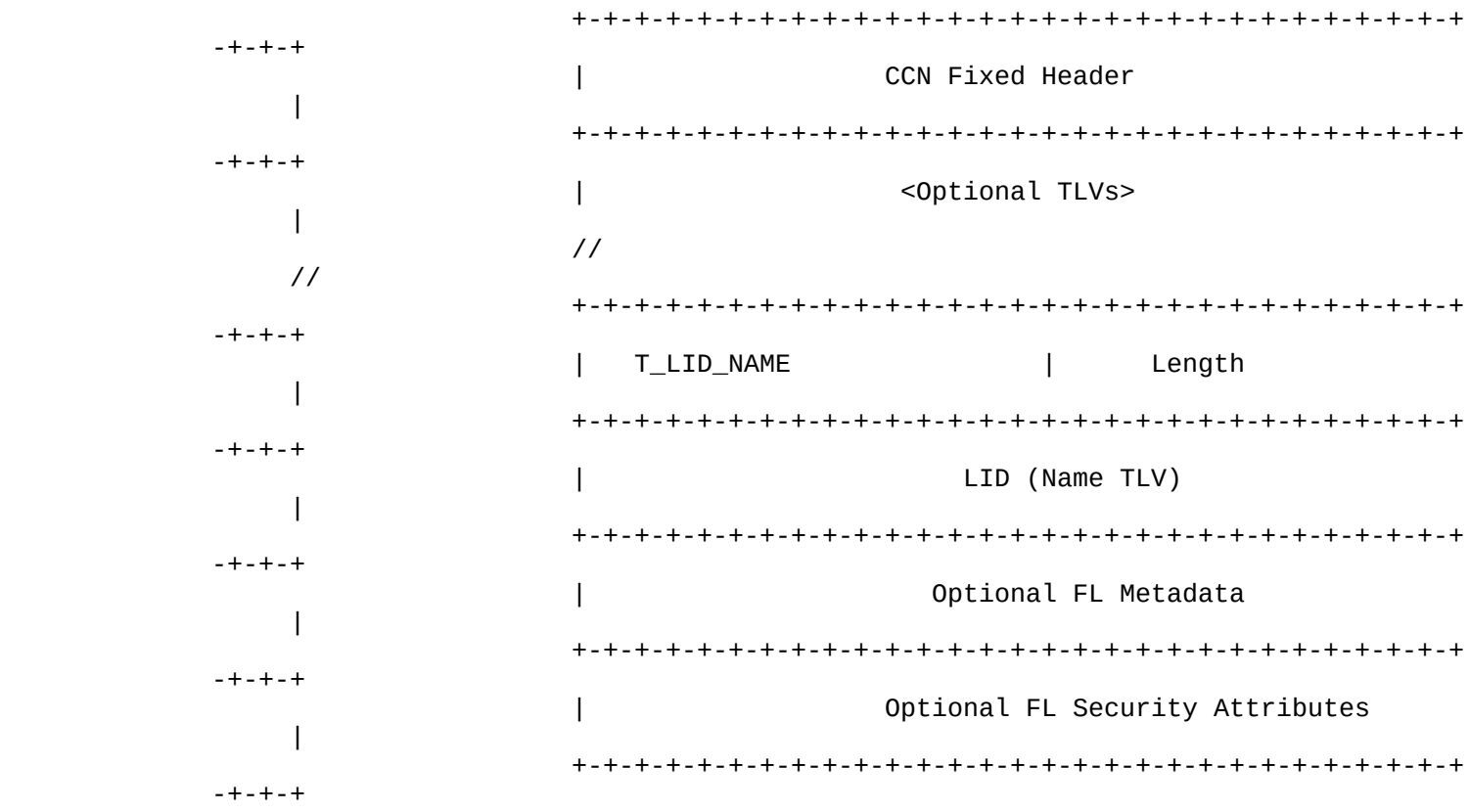


Figure 1: Optional TLV to include a Forwarding-Label Object

| Forwarding-Label | Meaning | Value |
|---|---|---|
| T_LID_NAME | Identifies an AS-ID/Gateway-ID/Host-ID | Name TLV |

Figure 2: Locater-Name Definition

## 4. Forwarding Label Processing Rules

The following discussion is based on the assumption that all forwarders must process optional header fields. In the context of CCN packet processing, FL object is relevant when the decision to forward the Interest message is to be made. At this stage, multiple options exist, assuming consistency of policy across the domain: 1) in a simpler scenario the rule may be that if a FL is included in an Interest message, then it should be given preference to the Interest

name. This is under the assumption that FL objects are trusted indirections included in the Interest message, which can be validated by the router if required; 2) in another scenario the forwarder could prioritize forwarding on the ID, and then forward on the LID at every hop; 3) in scenario where policy based routing is involved, more complex routing approaches can be considered at the network edge, such as the forwarder could apply service policy on the Interest ID

and choose to remove or swap with a new FL object irrespective of the FL object inserted in the Interest message, while the core nodes could use more simpler approach 1 or 2. Following are the steps when approach 1 is applied.

- During Interest packet processing when the forwarding decision is being made, if a LID is available then it should be preferred to the name in the Interest message for forwarding irrespective of feasibility of ID based routing.

- The validation of the FL depends on trust context. In trusted scenarios where the applications and network are managed by the same authority the forwarder can bypass validation. In untrusted scenarios the edge router may validate the FL send from the sender, and to avoid re-checks by successive forwarders these Interests can be marked to have been validated at the ingress point.

- If the lookup based on LID in the FL object succeeds then two possibilities exist: for non-terminating flows i.e. the LID FIB lookup results in a next hop and the Interest is forwarded ; for terminating flows, LID lookup invokes a service logic wherein the service either re-resolves the Interest ID to another LID hence a new FL object or removes the current FL object and subjects the Interest to regular processing based on the ID in the Interest message.

- If the FIB lookup based on the LID fails, then the router can try to forward it based on the Interest ID. If the latter fails, then the router could raise a error condition and feedback the message to the previous hop with appropriate error code.

-

## 5. PIT Processing Implications

To maintain simplicity of forwarding logic the purpose of FL object should be to guide the Interest to the producer or the closest source of the content/service, hence only be used for forwarding decision and not required for content object processing, however there may be usage scenarios where the FL state is required to be saved in the PIT and even piggybacked in the content object (CO).

In scenario when there is no binding between the ID and LID, and multiple Interests may arrive with different LID, then the expected outcome is to forward all such Interests with unique LID; in this case the PIT is required to save the LID along with the Interest ID and forward the duplicate Interest.

In another application it may be required to decouple the choice of one consumer's LID from another, i.e a secure binding exists between the ID and the LID. In this case, the PIT saves the FL object, and the returning CO should piggyback the Interest FL object and match it against the pending PIT entry before reverse forwarding. In case the FL object is swapped by intermediate routers, then the CO should be updated with the appropriate FL to ensure the PIT match the previous hops, these considerations are similar to those elaborated in [12].

## 6. Caching Implications

The considerations here follows from our previous discussion where the FL object is piggybacked in the CO as well. If there is a implicit security binding between the Interest ID and the LID then the FL object state is piggybacked along with the CO, and should be matched against the incoming Interest FL object before the cached content object is returned.

## 7. Multiple Domain Scenario

In wide area network scenarios, Interests cross multiple domains. If FL object is only trusted within domain boundaries, then the FL is removed before forwarding the Interest to the next domain, which then inserts a new forwarding label with associated security attributes at the ingress of the next domain. But if sufficient trust exists between domains to use the FL inserted by the previous domain, then the intermediate domains could avoid FL processing and use the FL passed on by the previous domains.

## 8. FL Security

FL object security is related to the purpose it is used for and the control plane mechanism used to manage them. Depending on the use case scenario of the FL appropriate security mechanisms should be applied to secure the control and data planes to avoid exploitation of this feature.

Generally, the major threats against the FL approach is to manipulate the relationship between the name and the FL. Such manipulations can happen in various scenarios, some of which are listed as follows: (i) a malicious interceptor (acting as a publisher) intentionally injects incorrect mapping into the name resolution system; (ii) the malicious interceptor (between the edge router and the resolution server) manipulates the mapping sent back from the name resolution system when the edge router queries the mapping system ; (iii) compromised intermediate routers maliciously change the FL, e.g., with the wrong FL object or out-dated FL object; (iv) untrusted application may inject invalid FL object in the Interest message.

Towards network level FL security, appropriate mechanisms should be applied to provide mapping provenance, mapping integrity and anti-replay attack to address these issues. The security mechanisms applicable to (i) and (ii) are similar to ones applied to secure other mapping systems such as LISP [5],DNS[7], (iii) requires new security mechanisms, one such way is to enable a domain level trust infrastructure so that the mapping between the name and the forwarding-label can be authenticated by successive routers.

In untrusted environments, when FL object is inserted in the Interest message from end hosts, appropriate authentication information should be included in the FL object to allow ingress routers to optionally validate the Interest ID to LID delegation [9]. Further, network could enable several policies, such as even to ignore the FL object, to handle FL objects from untrusted applications.

## 9. Use Case Scenarios

Here we provide the discussions related to using forwarding-label in different scenarios.

### 9.1. Handling Producer Mobility

In this application we discuss the use FL object to handle producer mobility using late-binding technique which is discussed in [8]. Here the mobile entity (ME) registers the persistent names which require mobility with its current point-of-attachment (PoA). The PoA then registers the mapping between the name and the PoA’s locator in its local name resolution system. Further the domain updates the ME’s home domain name resolution system with its current domain LID. When a correspondent nodes expresses Interest for the name, it is first resolved to the current ME domain by the home domain. When the Interest ingresses the domain, it is resolved again to the ME’s current location. Further PoA to PoA signaling can be enabled to enable seamless forwarding of Interests whenever ME changes its PoA.

### 9.2. Manifests

Manifests [6] may contain indirections to named content objects. In this case, FL object can be used to indicate its location while hierarchical or flat name ID map to the named object.

### 9.3. Interest Routing Optimization

Networks which hosts its own or third party content/service can benefit from the ability to handle Interest routing logic in its domain opportunistically. When Interests seeking a specific content

or service ingresses a network domain, the ingress router can redirect the Interest to the closest cache point or service location.

### 9.4. Routing Scalability

As discussed in [9], locator based routing can address routing scalability as the number of ASs are many orders less than the number of information objects. This reduces the forwarding table in the DFZ zone to order of number of AS in the Internet.

## 10. Informative References

[1] CCN Wire format, CCNX1., "http://www.ietf.org/id/draft-mosko-icnrg-ccnxmessages-00.txt.", 2013.

[2] Nikander, P., Gurtov, A., and T. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks", IEEE Communications Surveys and Tutorials, pp: 186-204, 2010.

[3] Atkinson, R., "An Overview of the Identifier-Locator Network Protocol (ILNP)", Technical Report, University College London, 2005.

[4] LISP, RFC6380., "https://tools.ietf.org/html/draft-ietf-lisp-sec-07.", 2014.

[5] LISP-Security, LISP-SEC., "https://tools.ietf.org/html/draft-ietf-lisp-sec-07.", 2014.

[6] CCNx, Manifest., "http://www.ccnx.org/pubs/draft-wood-icnrg-ccnxmanifests-00.html.", 2015.

[7] DNS-SEC, RFC4033., "DNS Security Introduction and Requirements.", 2005.

[8] Afanasyev, A., "Map-and-Encap for Scaling NDN Routing.", NDN Technical Report ndn-004-02, 2015.

[9] Azgin, A., Ravindran, R., and G. Wang, "A Scalable Mobility-Centric Architecture for Named Data Networking.", ICCCN (Scene Workshop) , 2014.

[10] Cisco System Inc., CISCO., "Cisco visual networking index: Global mobile data traffic forecast update.", 2009-2014.

[11] Zhang, Y., Zhang, H., and L. Zhang, "Kite: A Mobility Support Scheme for NDN.", NDN, Technical Report NDN-0020 , 2014.

[12] CCNx Label Forwarding, CCNLF., "http://www.ccnx.org/pubs/ccnx-mosko-labelforwarding-01.txt.", 2013.

[13] NSF FIA project, MobilityFirst., "http://www.nets-fia.net/", 2010.

Authors' Addresses

Ravishankar Ravindran
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ravi.ravindran@huawei.com

Asit Chakraborti
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: asit.chakraborti@huawei.com