Applicability and Tradeoffs of Information-Centric Networking for Efficient IoT
draft-lindgren-icnrg-efficientiot-03

Abstract

This document outlines the tradeoffs involved in utilizing
Information Centric Networking (ICN) for the Internet of Things (IoT)
scenarios. It describes the contexts and applications where the IoT
would benefit from ICN, and where a host-centric approach would be
better. The requirements imposed by the heterogeneous nature of IoT
networks are discussed (e.g., in terms of connectivity, power
availability, computational and storage capacity). Design choices
are then proposed for an IoT architecture to handle these
requirements, while providing efficiency and scalability. An
objective is to not require any IoT specific changes of the ICN
architecture per se, but we do indicate some potential modifications
of ICN that would improve efficiency and scalability for IoT and
other applications.

This document mainly serves as a problem statement and will not
present a conclusive architecture design. It can be used as a basis
for further discussion and to design architectures for the IoT.

Status of this Memo

material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Table of Contents

[1](#). Motivation

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. Depending on the context, the
IoT architecture may benefit from using an ICN or a host-centric
network (HCN). In practice, the right approach is a complex tradeoff
that depends on the applications and usage of the IoT network.

This document describes some advantages and inconveniences of using
an ICN for the IoT architecture, 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. Advantages of using ICN for IoT

   A key concept of ICN is the ability to name data independently from
   the current location at which it is stored, which simplifies caching
   and enables decoupling of sender and receiver. Using ICN to design
   an architecture for IoT data potentially provides such advantages
   compared to using traditional host-centric networks. This section
   highlights general benefits that ICN could provide to IoT networks.

2.1. Naming of Devices, Data and Services

   The heterogeneity of both network equipment deployed and services
   offered by IoT networks leads to a large variety of data, services
   and devices. While using a traditional host-centric architecture,
   only devices or their network interfaces are named at the network
   level, leaving to the application layer the task to name data and
   services. In many common applications of IoT networks, data and
   services are the main goal, and specific communication between two
   devices is secondary. The network distributes content and provides a
   service, instead of establishing a communication link between two
   devices. In this context, data content and services can be provided
   by several devices, or group of devices, hence naming data and
   services is often more important than naming the devices.

2.2. Distributed Caching

   While caching mechanisms are already used by other types of overlay
   networks, IoT networks can potentially benefit even more from caching
   systems, because of their resource constraints. Wireless bandwidth
   and power supply can be limited for multiple devices sharing a
   communication channel, and for small mobile devices powered by
   batteries. In this case, avoiding unnecessary transmissions with IoT
   devices to retrieve and distribute IoT data to multiple places is
   important, and storing such content in the network can save wireless
   bandwidth and battery power. Moreover, as for other types of
   networks, applications for IoT networks requiring shorter delays can
   benefit from local caches to reduce delays between content request
   and delivery.

2.3. Decoupling between Sender and Receiver

   IoT devices may be mobile and face intermittent network connectivity.
   When specific data is requested, such data can often be delivered by
   ICN without any consistent direct connectivity between devices.
   Apart from using structured caching systems as described previously,
   information can also be spread by forwarding data opportunistically.

3. Design Challenges of IoT over ICN

   As outlined in Section 2, there are potential benefits from using ICN
   to implement IoT communication architectures. However, in order to
   obtain a scalable and efficient architecture there are some aspects
   of ICN that must be specifically considered in making the right
   design choices for IoT. In fact, using an ICN may not be beneficial
   in all desired sub-functions and scenarios. This section outlines
   some of the ICN specific challenges that must be considered and
   describes some of the trade offs that will be involved. We will
   address these challenges in our proposed design choices later in
   Section 4.

3.1. Naming of Devices, Data and Services

   The ICN approach of named data and services (i.e., device independent
   naming) is typically desirable when retrieving IoT data. However,
   data centric naming may also pose challenges.

   o Naming of devices: Naming devices is often important in an IoT
      network. The presence of actuators requires clients to act
      specifically on a device, e.g. to switch it on or off. Also,
      managing and monitoring the devices for administration purposes
      requires devices to have a specific name allowing to identify them
      uniquely. There are multiple ways to achieve device naming, even
      in systems that are data centric by nature. For example, in
      systems that are addressable or searchable based on metadata or
      sensor content, the device identifier can be included as a special
      kind of metadata or sensor reading.

   o Size of data/service name: In information centric applications,
      the size of the data is typically larger than its name. For the
      IoT, sensors and actuators are very common, and they can generate
      or use data as small as a short integer containing a temperature
      value, or a one-byte instruction to switch off an actuator. The
      name of the content for each of these pieces of data has to
      uniquely identify the content. For this reason, many existing
      naming schemes have long names that are likely to be longer than
      the actual data content for many types of IoT applications.
      Furthermore, naming schemes that have self certifying properties
      (e.g., by creating the name based on a hash of the content),
      suffer from the problem that the object can only be requested when
      the object has been created and the content is already known, thus
      requiring some form of indexing service. While this is an
      acceptable overhead for larger data objects, it is infeasible for
      use when the object size is on the order of a few bytes.

o Hash-based content name: Hash algorithms are commonly used to name
   content in order to verify that the content is the one requested.
   This is only possible in contexts where the requested object is
   already existing, and where there is a directory service to look
   up names. This approach is suitable for systems with large data
   objects where it is important to verify the content.

o Metadata-based content name: Relying on metadata allows to
   generate a name for an object before it is created. However this
   mechanism requires metadata matching semantics.

o Naming of services: Similarly to naming of devices or data,
   services can be referred to with a unique identifier, provided by
   a specific device or by someone assigned by a central authority as
   the service provider. It can however also be a service provided
   by anyone meeting some certain metadata conditions. Example of
   services include content retrieval, that takes a content name/
   description as input and returns the value of that content, and
   actuation, that takes an actuation command as input and possibly
   returns a status code afterwards.

3.2. Efficiency of Distributed Caching

Distributed caching is a key opportunity with ICN. However, an IoT
framework must be carefully designed to reap the maximum benefits of
ICN caching. When content popularity is heterogeneous, some content
is often requested repeatedly. In that case, the network can benefit
from caching. Another case where caching would be beneficial is when
devices with low duty cycle are present in the network and when
access to the cloud infrastructure is limited.

However, using distributed caching mechanisms in the network is not
useful when each object is only requested at most once, as a cache
hit can only occur for the second request and later. It may also be
less useful to have caches distributed throughout ICN nodes in cases
when there are overlays of distributed repositories, e.g., a cloud or
a Content Distribution Network (CDN), from which all clients can
retrieve the data. Using ICN to retrieve data from such services is
beneficial, but in case of dense occurrence of overlay CDN servers
the additional benefit of caching in ICN nodes would be lower.
Another example is when the name of the data has a different meaning
depending on the context, or if the name refers to an object with
variable content/state. For example, when the last value for a
sensor reading is requested, the returned data should change every
time the sensor reading is updated. In that case, ICN caching may
increase the risk that cache inconsistencies result in old data being
returned.

3.3. Decoupling between Sender and Receiver

Decoupling the sender and receiver is useful mechanism offered by the
ICN approach, especially for content retrieval with duty cycling
devices or devices with intermittent connectivity. However, in order
to efficiently retrieve data it must be possible for requestors
(receivers) to easily deduce the name of the data to request, without
any direct contact with the responder (sender).

Nevertheless, de-coupling is a challenge when authentication is
needed for management and actuation, or when real-time interaction
between devices is necessary. Solutions for object security
supporting decoupled authentication (e.g., similar to signing by
proxy), and solutions for pushing data to decoupled entities must be
explored.

[4](#). 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.

[4.1](#). 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 4.12](#).

[4.2](#). 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 requestor, 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.

4.3. 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.

4.4. 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 4.2, 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 4.7, 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 4.8 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 requests act as one-time subscriptions. 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.

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.

4.5. 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 4.8) 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 4.4. 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.

4.6. Decoupling and roles of senders and receivers

Since ICN networks essentially support a request/response model of
interaction, we denote the receivers of information as requestors,
and the senders of information as responders. The ICN network in
itself provides decoupling of requestors 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 requestors.

   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 requestor 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
   requestors (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 requestors to ultimate
   responders may go through many IoT processing nodes that collect,
   process and re-publish data through an ICN for various purposes.

4.7. 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 requestors 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 requestor, 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 requestors. 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 4.3.
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.

4.8. Capability advertisements

Capability advertisements and discovery can be used by requestors 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 4.7.

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

   As described in Section 4.2, 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 requestors 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 requestors 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)

4.10. What's naming and what's searching

   As described in Section 4.2, 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.

4.11. 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 requestor
   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.

4.12. 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 4.4. 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.

4.13. 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.

5. 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.

   Object Security is based on the idea of securing information objects
   unlike session-based security mechanisms which secure the
   communication channel between a pair of nodes. This reinforces an
   inherent characteristic of ICN networks i.e. to decouple senders and
   receivers. In the context of IoT, the Object Security model has
   several concrete advantages. As discussed earlier in Section 2.1, in
   many IoT applications data and services are the main goal and
   specific communication between two devices is secondary. Therefore
   it makes more sense to secure IoT objects instead of securing the
   session between communicating endpoints.

   It is important that while security mechanisms complement the ICN
   architecture in a coherent fashion, they do so without laying down
   any strict requirements or constraints. Therefore, the decision of
   what security mechanisms are employed should be handled at a layer
   above ICN, in this case within the IoT framework. However, the ICN
   layer should not be completely oblivious of Object Security. At this
   point it is important to distinguish between the different aspects of
   Object Security i.e. integrity, authenticity and confidentiality.
   ICN provides data integrity through Name-Data Integrity i.e. the
   guarantee that the given data corresponds to the name with which it
   was addressed. Typical ICN protocols provide Name-Data integrity
   using various schemes such as hash-based names and signatures.
   Signature-based schemes additionally provide data authenticity.
   Otherwise data authenticity should be provided in layers above the
   ICN layer. Data confidentiality should also be handled above the ICN
   layer. This facilitates flexibility and allows IoT applications more
   freedom to decide which encryption scheme suits them best (session-
   based encryption, object-based encryption or a hybrid).

   Though the idea of Object Security is very much in line with the ICN
   concept, there can still be some use cases where Object Security does
   not add much e.g. a Pub/Sub interaction where a client is expected to
   interact more or less with the same server node (a session-based
   security protocol should suffice here) or use cases where application
   layer headers should also be secured (which can be achieved by TLS/
   DTLS). We, therefore, effectively imply that there is no need to
   modify typical ICN standards to accommodate Object Security in its

entirety.

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

5.1. 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.

5.2. 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.

5.3. 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.

6. Acknowledgements

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: