University of Amsterdam

Faculty of Physics, Mathematics and Informatics

MSc System and Network Engineering

Research Project 1

Feasibility of Internet-of-Things (IoT) applications in Information-Centric Networking

Abstract

Currently the Internet is host-centric network based on the end-to-end principle. Information-centric networking (ICN) is an approach to radically change the Internet by using a data-centric approach where not the host has an unique address but the data object itself. Making it much easier to process data along the way or the aggregate request/answers, improving the efficiency of the Internet. In this paper we looked at how Content Centric Networking (CCN) compares to Named Data Networking (NDN) with regards to Internet-of-Things (IoT) applications. We found that for the most part CCN and NDN are fundamentally very similar, however, there are some small differences. For our use case, we found that NDN is a better choice than CCN. Mainly because NDN does not need to have exact name matching, which makes it possible for us to send data upstream. Sending data upstream is necessary to solve the routing challenges we had in our use case.

Jenda Brands *Olaf Elzinga*

July 3, 2016

Contents

1 Introduction

The Internet of today is a host centric network which is based on perpetual connectivity and the end-to-end principle. ICN is an approach to change the Internet architecture into a data centric network, meaning that data becomes independent from location, application, storage, and means of transportation, enabling in-network caching and replication [7]. The expected benefits are improved efficiency, better scalability with respect to information/bandwidth demand and better robustness in challenging communication scenarios [7]. Currently there are several projects which aim to implement this architectural concepts including: Content Centric Networking (CCN), Named Data Networking (NDN), Network of Information (NetInf), Data-Oriented Network Architecture (DONA) and many more.

CCN was created by the Palo Alto Research Center (PARC) in 2007 and in 2009 they first released a software implementation. The NDN project started in 2010 and is funded by the National Science Foundation (NSF). The NDN project originally used CCN as its codebase, but as of 2013 has forked a version to support the needs specifically related to the NSF-funded architecture research and development (and not necessarily of interest to PARC) [16]. Since both projects have gone their own way, the protocols are, at this time, incompatible [23]. As both projects are highly related but with certain differences, it is interesting to understand where these differences come from and what these differences mean in practice. In this paper, we look at the advantages and shortcomings of CCN and NDN with regards to Internet-of-Things (IoT) applications. By looking at both with a practical and theoretical view, we evaluate how both projects are evolving and how to evaluate the differences which have emerged after the fork.

The main research question is: *'How does Content Centric Networking compare to Named Data Networking with regards to IoT applications?'*
This question can be divided into the following sub-questions:

1. What are the differences between CCN and NDN?
2. Are there improvements to be made to make CCN and/or NDN more accessible for IoT?
3. Are there IoT applications which benefit more from CCN then from NDN and vice versa?

The rest of the paper is organized as follows: in Section 2 we will discuss the related work. In Section 3 we will describe our use case. In Section 4 we will look at how both CCN and NDN work and compare them. In Section 5 we look into architectures we think are possible and choose one of them to make a proof of concept in Section 6. This paper ends with a conclusion and discussion of future work.

2 Related work

Currently, there is a lot of research going on in the field of ICN. Most of the research focuses on the challenges described in the Internet draft "draft-irtf-icnrg-challenges-06" [8]. This draft describes challenges such as: naming, data integrity, security and routing. More interesting for our research project is the Requirements and Challenges for IoT over ICN draft [26]. The survey of Saxena et al. [19] studies the NDN architecture and various schemes proposed for its different characteristic features like: naming, adaptive forwarding and routing, caching, security, mobility etc.

Lindgren et al. [10] looked at some of the benefits and trade-offs associated with using ICN technology over IP for an IoT scenario. The authors of the paper 'Information centric networking in the IoT' [3] carried out experiments with NDN on a real IoT deployment consisting of tens of constrained nodes in multiple rooms of multiple buildings. Their work showcased that ICN is indeed applicable in the IoT, and that it can offer advantages over approaches based on 6LoWPAN/IPv6/RPL in terms of energy consumption, as well as in terms of the RAM and ROM footprint.

In the paper by Zhong Ren et al, a light-weight version of CCN is proposed by the name of CCN-WSN [18]. This CCN variant is designed for wireless sensor networks, mainly for constrained devices. As this paper was written in 2013, when CCN was still at a version below 1.0, most of the improvements proposed are not applicable anymore because CCN version 0.x and 1.0 differ too much in fundamental aspects. Also CCN-WSN was proposed as a light-weight implementation of which constrained devices should benefit. As the use case defined in this research makes use of sensors placed in cars, these are not considered constrained devices in terms of power of bandwidth.

3 IoT use case enabled by ICN

For this project, a specific IoT use case is defined in order to compare CCN and NDN. By looking at how this use case can be implemented using CCN on the one hand and NDN on the other hand, several features of the techniques might be identified as well as any shortcomings. From a more generic point of view, problems in using either techniques can be identified as well. As the use case itself is very specific it does not cover IoT as a whole, but just a certain category within IoT.

3.1 Use case definition

The use case created in this research involves a highway where cars (and other vehicles) are driving. All cars are fitted with several sensors that produce data like: (GPS) location, speed, temperature et cetera. The cars are able to send this data to a (not further specified) wireless network.

The aim is to retrieve certain data about the road, for which sensors function as a source. This could be the average speed within a certain sector, or the average outside temperature of a part of the road. In order to get this information, at least two different ways can be thought of to retrieve this data, namely:

- Getting data from sensors in the road itself
- Getting data from sensors in the cars on the road

3.1.1 Current measuring methods

Currently it is already possible to measure the average speed of cars on a certain place on the road. This is done through induction loops that are installed in the pavement [4]. These loops are connected to electronics on the side of the road, which are able to send the measurements to a central place.

In this use case, the same goal is achieved by using sensor data from cars. However, advantages appear when using data from the cars, instead of using data from induction loops in the pavement.

As induction loops need to be installed in the pavement, its use is restricted to a specific location. Whenever an induction loop fails, measurements on that location are not possible anymore. Costs will increase as well with the need for measurements on more locations, as each location needs an induction loop.

Instead of very specific locations on the road that could be monitored by induction loops, car sensors allow any part of the road to be monitored as there is no dependency on physical sensors in the pavement. It would also be possible to monitor a range on the road i.e. measure the average speed from point A to point B.

3.1.2 Research scope

Our research focuses on the second way to retrieve the data, by using sensors in the cars and not data from sensors in the road itself. Figure 1 graphically shows the concept of the use case. Multiple cars are driving on the highway. The highway itself is divided into multiple sectors, which is graphically shown by the different outline styles around the highway. In order to find out the average speed in one sector, all cars in that sector at that moment could be polled for their current speed, after which the average of the speeds is calculated. The same holds for temperature sensors for example.

This use case does imply some requirements and constraints on the following subjects:

- Network connection
- Naming
- Routing
- Push based communication
- Data aggregation

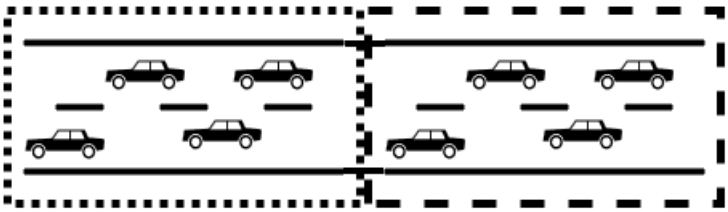


Figure 1: Highway use case

Network connection

For the cars to be able to send Interest and Data packets to the network, a connection to a network is needed. This connection can be made over the existing mobile networks (2G, 3G or 4G) or by creating a whole new separate network specialized for this purpose. As the underlying layers are somewhat out of scope for this research, not much emphasis is put on the underlying network technologies. For now, in this paper the assumption is made a separate network is available for all car sensors.

Naming

As the ICN paradigm describes, names are used to address pieces of data rather than addressing specific hosts where the data is stored. When deciding to go for a data centric approach, the naming scheme to address the data becomes important. For this use case it is essential to come up with a naming scheme that very specifically describes what type of data it addresses. As will be shown later on in the paper, some naming schemes can and some can not work with our use case. A basic structure for the naming scheme was defined, which starts with the physical location, followed by the device (car or access point) and then the actual sensor data. In practice this naming scheme looks as shown in Listing 1.

```
/nl/tno/A10/S37/ap/7-ABC-44/82k
|   |   |   |   |  |         |
|   |   |   |   |  |       Speed
|   |   |   |   |  License plate
|   |   |   |   Access point ID
|   |   |   Sector ID
|   |   Highway number
|  Company
Country
```

Listing 1: Possible naming structure for our use case

For this use case it could come in handy to be able to publish data on multiple names, or vice versa: having multiple sensors publishing on the same name. In this case all cars could be publishing data on a generic name for speed (or temperature) in a certain location. Also, if one sensor could publish data on multiple names, it would become possible to publish data on a car specific name address, as well as publishing the same data on a location specific address.

Routing

Just as a network connection is needed in order for the cars to send their sensor data, routing is essential as well for data to be sent across the network. As cars are constantly moving, one of the routing challenges is getting an application to send Interest packets to car sensors.

Push based communication

As ICN is designed in a pull-based fashion, it might become hard to have any push-based support within ICN. For this use case it could be a great plus to have push-based support, as this would mean a car could send its data to a central point within the network, instead of being pulled for the data. Discovery of cars in a certain sector would also benefit from push-based communication, as cars would be able to register itself whenever they change sectors.

Data aggregation

In order to calculate average speeds, or an average temperature, it is important to have a set of data instead of just one measurement. For the network it would be nice to have data aggregation possibilities including ways to calculate average speeds. If the network does not support any form of aggregation, this should be done on application level.

4 Theoretical Comparison

In this section we present a brief overview of the inner workings of ICN. We will start with an overview of the concept of ICN, followed by a more detailed description of several aspects in ICN. Furthermore, the CCN and NDN implementations of ICN are discussed and compared based on the aspects discussed.

4.1 The philosophy of ICN

The current Internet follows a host-based approach. This might be considered strange since one is interested in a certain piece of information, rather than a host. When someone visits a web page, that person is interested in content on that website rather than the host where the website resides. ICN is content based, meaning that a person requests the website, without having to know where the actual content is stored. This is for the network to sort out.

Nowadays the Internet is often described as being an hourglass architecture, where the thin waist of the hourglass resembles the network layer, IP in this case. The network layer forming the waist of the hourglass is transparent enough, so that almost any application can run on top of it, and simple enough, so that it can run over almost any link-layer technology [24]. Figure 2 shows the hourglass example with the current Internet versus ICN.

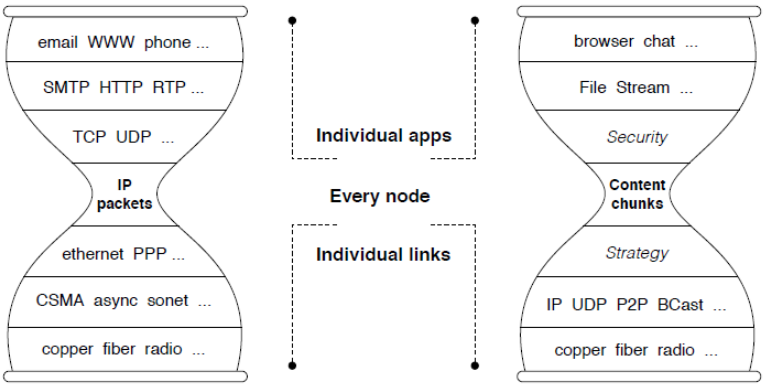


Figure 2: Internet hourglass vs ICN hourglass

Source: NDN Consortium [25]

Figure 2 shows how ICN compares to the current Internet architecture. The content chunks in ICN are situated on a higher level than IP packets in the Internet. This means ICN can run on top of IP, as well as other technologies.

One of the differences with ICN and the current Internet is that ICN is a completely pull-based architecture. Whenever an Interest is sent, a Data packet is expected back. The concept of ICN does not allow for any push-based communication.

4.2 Inner workings

The keystone of ICN are data objects. In CCN this object is called a content object (CO) and in NDN a named data object (NDO), for the simplicity we will call them CO. A CO can be any kind of object, for example: videos, web pages, documents and photos, as well as streams and interactive media. The CO is independent from location, application, storage and means of transportation. When a CO is used at a global scope, the CO must also have a globally unique name. To get a certain object, ICN uses two types of packets: Interest packets and Data packets. A requester sends out an Interest packet for specific data, which is forwarded by nodes into the network. When the Interest packet reaches the right node, where the data currently resides, a Data packet will flow back using the same path the Interest packet took.

Each CCN/NDN node maintains three data structures, namely: the forwarding information base (FIB), the pending interest table (PIT) and the content store (CS). The FIB, which contains routing information that comes from local applications prefix registrations and routing protocols [1]; The PIT keeps state of forwarded Interest packets which have not (yet) been answered with Data packets; The CS temporarily caches incoming Data packets to enable delayed or later requests to be handled faster and more efficient. Figure 3 illustrates the basic concept of CCN and NDN; When consumer 1 wants an object which resides at the producer, it sends an Interest packet to the router, which stores the Interest in its PIT and forwards it to the producer. In this case the producer replies to the Interest packet from consumer 1 with a Data packet, which is sent to the router. The router caches it and forwards it to consumer 1. Now consumer 2 wants the exact same object and sends an Interest packet to the router. The router then sees that the object which consumer 2 is interested in, is cached in its CS and decides to answer the Interest packet with the cached data. Besides caching, ICN also tries to aggregate data, for example, if both consumer 1 and 2 send out an Interest packet for the same object, the router will then send a single Interest packet to the producer and remember in its PIT table which nodes are interested in the object. When the routers receive the Data packet(s) from the producer, it will forward the packet(s) to all nodes interested.

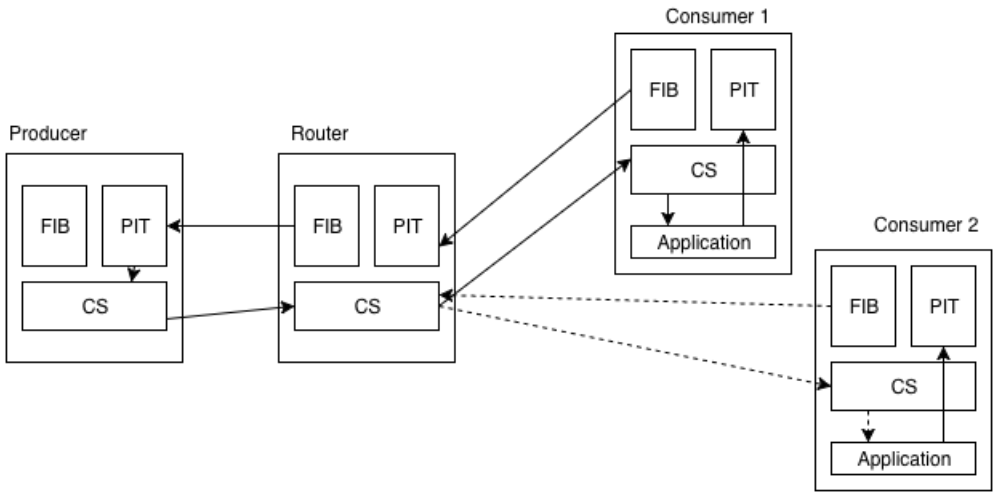


Figure 3: Example of caching and Interest aggregation in NDN

4.3 Naming

In ICN two naming schemes are proposed: flat names and hierarchically structured names. Although it is possible to route flat names [5], it is the hierarchical structure of the Internet today which makes it as scalable as it is. The NDN design assumes hierarchically structured names [17] as well as the CCN draft which states that each name should be hierarchical [13]. Just like every other hierarchical naming structure, the name starts with an empty root label followed by non-empty labels. Each label is separated by a single slash, for example:

/uva/fnwi/sne/images/icnlab/png/date/20160603/res/500/800

Both CCN and NDN use the same structure and divide the content name into three segments [11]:

1. **Globally routed segment**: The first part of a hierarchical name is used to make the name globally routeable, this is necessary because a global network like the Internet needs to be scalable, otherwise the routing tables would become too big and look-ups of routes would take too long.
2. **Application segment**: The next part of the hierarchical name is assigned to an object, for example an image or web page.
3. **Protocol specific segment**: The last part of the name consists of attributes related to the protocol. These can be anything like: content-version information, chunk number, date/time stamp and content specific information (e.g. resolution and video codec).

Figure 4 illustrates the three segments used in hierarchical naming. CCN uses only exact name matching [21], meaning that Interests only match with exact matches. On the other hand NDN uses the old ccnx name matching, which doesn't have to match exactly. For example, an Interest for /nl/uva/icnlab/image will match the content /nl/uva/icnlab

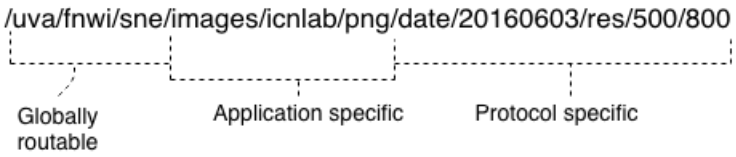


Figure 4: Example: hierarchical name structure

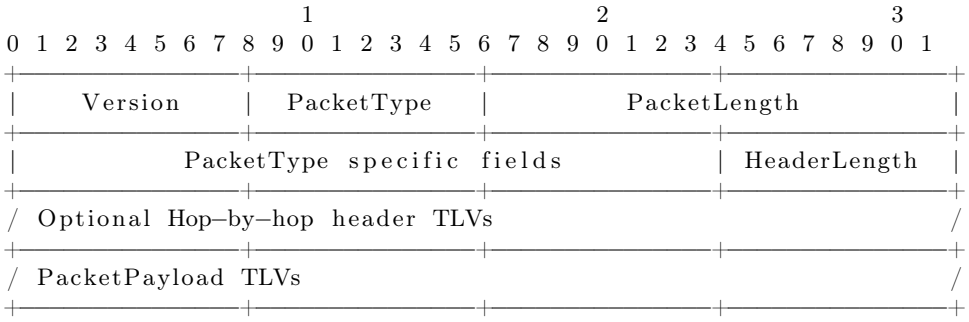
4.4 Packet format

This section describes the packets used in CCN and NDN.

4.4.1 CCN packet format

The overall packet format in CCN is shown in Listing 2. The default packet contains a version field, a field for the packet type (Interest or Content Object) and the total length of the packet including all headers. Then type specific fields are available, depending on the type of packet. Also the Header length is set. At last some Hop-by-hop header TLVs can be set, which can be

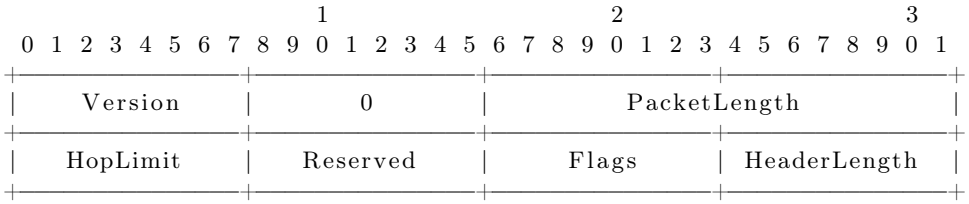
used for Interest lifetime, Content cache time etc. At last the packet payload TLVs are placed, for the actual payload.



Listing 2: CCN overall packet format
Source: CCNx Messages in TLV Format [12]

Interest packet

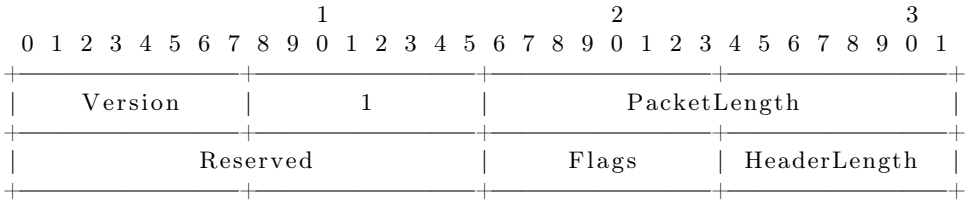
The Interest packet format used in CCN is shown in Listing 3. The header is similar to the generic header discussed before, with the difference that the PacketType field is set to zero, indicating this is an Interest packet. Also the PacketType specific fields are defined here, starting with a field for the HopLimit, followed by a reserved field which should be set to zero and as there are no flags available yet this field should be set to zero as well [12]. The PacketPayload consists of at least the CCN message.



Listing 3: CCN Interest packet header
Source: CCNx Messages in TLV Format [12]

Data packet

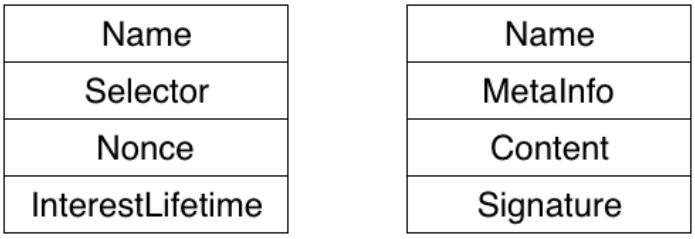
The Content object packet format in CCN is shown in Listing 4. This header is very similar to the Interest packet header, although the PacketType field is now set to one, indicating this packet is a CO. The PacketType specific fields consist of a reserved field and a field for flags. However, since also for the Content Object packet no flags are defined, this should be set to zero.



Listing 4: CCN Content Object packet header
Source: CCNx Messages in TLV Format [12]

4.4.2 NDN packet format

Each NDN packet (both Interest or Data) uses Type-Length-Value (TLV) encoding, which has no fixed size and thus can be extended or reduced depending on the requirements of the network. The elements NDN uses for both Interest or Data are illustrated in Figure 5.



(a) Interest packet format (b) Data packet format

Figure 5: NDN packet format

Interest packet

The Interest packet starts with the name. An NDN Interest packet can contain a set of different Selectors, these elements are optional and used for further qualifying Data that may match the Interest. Selectors that are currently available are [1]:

- MinSuffixComponents
- MaxSuffixComponents
- PublisherPublicKeyLocator
- Exclude
- ChildSelector
- MustBeFresh

The Nonce is a 4-octet long byte-string which in combination with the Name should make an Interest packet unique and is used to prevent loop in the network. And finally, the InterestLifetime indicates the time before an Interest times out in milliseconds, by default the InterestLifetime value is 4000.

Data packet

Just like the Interest, the Data packet contains the Name of the content. Where after, MetaInfo follows which consists of three optional elements: the ContentType, FreshnessPeriod and FinalBlockId. Where after the content follows and at the end the digital signature is placed which consists of a digital Signature of the other three elements (Name, MetaInfo, and Content), the SignatureType, and the KeyLocator.

[1] http://named-data.net/doc/ndn-tlv/interest.html

4.5 Forwarding

ICN is designed in a pull based fashion, meaning that a node has to send an Interest message in order to receive data. This is unfortunately not ideal for a lot of applications such as chat, email and voice. The authors of [22] talk about an NDN email application with a similar architecture how it is used in the Internet today, where mail agents push the mail to other agents until it reaches the right mailbox. They used an initial Interest message to force the other side to send an Interest message to pull the mail.

The rest of this section covers how both CCN and NDN handle forwarding of Interests and Data messages.

4.5.1 CCN Interest forwarding

According to the information stated in [14], the Interest forwarding process works as follows. As soon as an Interest packet arrives at the CCN forwarder, the first check done is on the hop limit of the Interest. If the Interest came from a remote host and the hop limit is zero, it will drop the Interest. If the hop limit is larger than zero, it will decrement the hop limit. Next it will first try to match the Interest with the CS if available. If the Interest matches a content object in the CS, the forwarder will return that object to the previous hop (where the Interest originates from) and discard the Interest. If no CS is available or no match occurred in the CS, the Interest will be matched against the PIT. If the Interest matches an entry in the PIT, the forwarder will aggregate the new Interest with the one in the PIT, and won't forward the Interest any further. An important note here is that whenever the Lifetime of the new Interest is greater than the Lifetime of the one already in the PIT, the forwarder needs to re-forward the Interest after the current Lifetime expires. Also the MinimumPathMTU of the new Interest should not exceed the MinimumPathMTU of the Interest already present in the PIT. If the Interest does not match anything in the PIT, a new PIT entry is created. When the HopLimit equals zero, the forwarder will only forward to local apps. If the HopLimit is higher than zero, the forwarder will retrieve a list of possible next hops, thereby excluding the ingress hop, as that is where the Interest came from. The forwarding strategy is applied and the Interest is forwarded to the next hop, unless the forwarding strategy results in an empty set. In that case the PIT entry is removed. A graphic representation of the process described above can be found as a flowchart in Appendix A.1.

4.5.2 CCN Content Object forwarding

According to [14], whenever the forwarder receives a CO the forwarder will first check the PIT to find all entries that are satisfied by the CO. If this results in no entries, the forwarder will drop the CO. If the CO is not dropped (meaning at least one PIT entry matched), a check is done if the CO originated from an expected previous hop. The previous hop should be in the FIB of the forwarder, otherwise the CO should be dropped. This makes sense since all Interests are sent out according to the FIB. If suddenly another host replies without ever being sent an Interest, this would be strange. When the CO is still not dropped, the CO is forwarded to all previous hops denoted in the matched PIT entry's ingress set. If a CS exists on the forwarder,

it will save the CO in the CS. A graphic representation of the process described above can be found as a flowchart in Appendix A.2.

4.5.3 NDN Interest forwarding

When a node receives an Interest message, the first check is if the localhost scope has been violated. Where after, the Name and the Nonce are checked against the Dead Nonce List. Normally if an Interest is unsatisfied after the InterestLifetime expires, the PIT entry is deleted. However, when a short InterestLifetime is used, this could introduce loops in the network, therefore, the Dead Nonce List was introduced to keep track of deleted PIT entries and keep the size of the PIT manageable. To reduce memory usage, the Interest Name and Nonce are stored as a 64-bit hash [20]. It could be the case that non-looping Interests could be considered looping (false positives), but the probability is small, and the error is recoverable when the consumer retransmits the Interest with a different Nonce [20].

Now, when a node receives an Interest message which is already in the Dead Nonce List, the message will be discarded. When there is no match with the Dead Nonce List, a PIT entry is created.

Before the Interest is processed any further, it will again be checked against the Dead Nonce List and the Nonce in the PIT entry is also verified. If a match is found the packet can be either a loop or multi-path arrival and therefore will be dropped. If the Interest is not pending it is looked up in the CS, if it is a hit the Interest will be replied with the content; if not, the

4.5.4 NDN Content Object forwarding

When an NDN node receives a Data message, it will first check if the localhost scope has been violated. Then NDN checks if there is a PIT entry patching the Data message. If so, the data is added to the CS, if not in most cases the Data packet will be dropped. However, one can configure a NDN node to cache unsolicited data if needed. The current implementation allows any unsolicited Data packet to be cached if this Data packet arrives from a local Face [1]. Then for each Interest the nonce (if needed) is added to the Dead Nonce List. Where after, the PIT entry is marked as satisfied and for each Interest, the data will be forwarded to the face where the received the Interest from.

4.6 Routing

Compared to other parts of CCN and NDN, routing is still in a very early research stage. The authors of [6] introduced Distance-based Content Routing (DCR). DCR is the first approach for name-based content routing within the ICN domain. DCR does not require any routing information about the physical topology of the network or information about all the replicas of the same content to provide multiple shortest paths to the nearest replica of content, as well as to all or some instances of an NDO or name prefix. The authors of [2] sketch out a design that makes NDN routing scale as well as today's IP Internet, potentially significantly better. This can be done by applying the well understood concept of map-n-encap to the context of NDN so that only ISP name prefixes need to appear in the global routing table.

Currently there are no routing implementations for CCN, however, for NDN there is, namely Named-data Link State Routing (NLSR). NLSR is an intra-domain routing protocol for Named Data Networking [9]. It runs the following simple extension of Dijkstra's algorithm to produce multiple next hops to reach each node. It first removes all immediately adjacent links except one and uses Dijkstra's algorithm to calculate the cost of using that link to reach every destination in this topology.

Both CCN and NDN use the Routing Information Base (RIB) to store static or dynamic routing information registered by applications, forwarding daemon and routing protocols (e.g. NLSR). The routing information in the RIB is used to calculate next hops for FIB entries in the FIB.

4.7 Differences

We found that both CCN and NDN are fundamentally very similar, however, there are differences how these fundamental principles come to life in each project.

The naming is very similar except for the fact that a CCN name must be an exact match, where NDN names do not have to be an exact match. Both projects use a different packet format. CCN describes a fixed-size header format, whereas NDN uses a variable-size header. Even though both CCN and NDN make use of the TLV format which makes extension of packets possible. When we look at forwarding, an interesting difference was identified. When a CCN forwarder receives an Interest packet, it will first check in the CS if a CO is present in the CS which satisfies the Interest. If there is no CO in the CS that satisfies the Interest, it will create a new PIT entry and ultimately forward the Interest. NDN takes another approach by first checking the PIT if an Interest is already pending. Afterwards the CS is checked. This is done in this specific order to reduce lookup overhead. The assumption by the NFD team (NDN) here is that probably the PIT will be significantly smaller than the CS [1].

We looked at the possibility of having multiple sensors publishing on the same name. Meaning that multiple COs have the same name. This could only work and be useful in very specific use cases. Unfortunately, this does not work well for our use case. Whenever an Interest packet is sent for this single name, the first object closest to the source of the Interest would satisfy the Interest. Practically this creates a sample taken at random, however the randomness is highly doubtful. This solely depends on which producer is closest to the source of the Interest.

Within CCN no routing is implemented yet, except for static routing. NDN already features NLSR, a link state routing protocol. Also static routing is possible with NDN. In the near future, CCN will probably feature routing protocols as well as there is already some research done in this field [15], but up until this day no routing protocol is publicly available.

The last main component we looked at, was if it was possible to send data within an Interest message. This turned out to be not possible, as the field within an Interest message can not be used for any additional data. One way to send data though, is to place it in the name by using longest prefix matching.

One of the main differences we benefit from for our use case is the possibility to push data. Since ICN does not provide any push-based communication by design, we tried to find another way of being able to send data in an Interest message. In NDN this can be done by the use of

longest prefix matching, which is not possible in CCN. This difference makes NDN suitable for our use case, whereas CCN can not be used due to the need of exact name matching.

Table 1 shows an overview of all differences discussed above.

Table 1: Comparison of important features in CCN and NDN

	CCN	**NDN**
Naming	Must match exactly	Longest prefix matching possible
Packet format	Fixed-size header	Variable-size header
Forwarding	CS check first, PIT Entry second	PIT Entry first, CS check second
Single name multiple objects	Likely to cause issues	Likely to cause issues
Routing	No routing implemented yet	Link state and static
Push-based possibilities	No longest prefix matching	Longest prefix matching
Data in Interest	Not possible	Not by design

5 Possible architectures

In this section, we will present four possible architectures we designed to solve the use case. We will look at the possibilities of the designs and how they will fit into the ICN philosophy. In the end we will argue which of the architectures we think fit the most within the ICN paradigm and which not.

To transfer sensor data from the car to the access point, one can make a pull-based or a push-based design. The main challenges for pull-based protocols is finding the path to the content. In a regular sensor network this wouldn't be a problem since most sensors do not change position very often. However, in our use case, the sensors will constantly change position. Therefore, traditional routing protocols are not able to converge fast enough to reliably route between the sensor and the other devices in the network. The main challenge for push-based protocols is the fact that both CCN and NDN don't support native push-based protocols since it only support pull-based traffic.

5.1 Different architectures

Architecture 1: Multicast

One of the simplest network designs is to use multicast traffic to reach the sensor. If a sensor joins a network, it would automatically join the multicast group where other CCN/NDN nodes listen to. When a sensor receives an Interest message, the Data message will automatically find its way back based on the PIT entry, therefore, no static route or routing protocol is needed. The downside of using multicast is that the same name needs to be used for each car, thus making it impossible to use unique names. Secondly, the first Data packet that arrives at the access point will satisfy the PIT entry, meaning that all data arrived after that moment will be discarded, unless it is configured differently.

Architecture 2: Longest prefix matching in Interest

Since there is no possibility to send data in an Interest message, one can choose to use more labels than necessary to match the name of the collection point. The extra labels can be used to send small variables such as: speed, temperature, or license plate. For example, the Interest for the access point on sector 37 could be /nl/tno/A10/S37/ap. Then we add a label with the license plate (7−ABC−44) and a label with the current speed (82 km/h), which will result in the name: /nl/tno/A10/S37/ap/7−ABC−44/82k. Longest prefix matching would be a solution for an IoT application which only has to send small amounts of data. However, when more data needs to be sent longest prefix matching would be inappropriate. Although in theory it maybe possible to send a large amount of data inside Another argument for not using this method is that the name is not supposed to contain data. CCN for example, does not support this method since names have to be an exact match.

Architecture 3: Centralized collection

Centralized collection is, out of the four architectures, the one which has most in common with IP-based solutions. The car communicates directly with a central server within the network. This means that there could be N_x nodes between the application that collects the sensor data and the sensor data itself. The central server would serve multiple highways throughout the

country, resulting in a relatively low amount of servers needed to process the sensor data. The car should have a static default route (e.g. /nl/car) to the next hop (access point) it is connected with. When a car sends an Interest message to the application on the server, the message will travel through the backbone. Because of the PIT entries on all forwarding nodes it finds its way back. However, when data needs to go upstream there opens up a huge routing problem since the application can not use the PIT to find its way to the car. Thus routing protocols are needed, but because cars can change from access point regularly, traditional routing options are not able to converge fast enough. Mainly because neighbor establishment and the propagation of routes after that take some time. Besides that before a route is declared dead, the car could be connected with a different access which could introduce routing problems.

To find a solution to this problem, routes could be aggregated. For example, in the case that one sector has only one access point, it is possible to route "/nl/tno/A10/s38" all traffic for that sector to that access point. However, this design limits the amount of access points used per sector. When for example multiple access points are used, the router that connects these access points to the backbone could not know at which access point a specific car is unless it has the route to the car itself. This can be solved by using static routing along the way, which makes this strategy very complex. Another option would be to create a static route between the application and the car. However, now we rely on an IP stack to route traffic from the car to the application instead of ICN. For example, in the case that one sector has only one access point, it is possible to route all traffic for a sector with name "/nl/tno/A10/s38" to that access point.

Architecture 4: Subscription based

When a car connects with a new access point, the car will send an initial Interest with the unique name of the car and the network address. With this information the access point will add a static route to reach the content of the car and send back an Acknowledgement message to confirm the registration. Every x seconds the access point will pull the sensor data from the car that is currently registered. When a car does not respond to an Interest message, the car will be removed from the list and will not be pulled anymore. The content of the data the car produces has a short expiration time, therefore, the relevance of the data could be guaranteed. The access point, on its turn, publishes the data of all the cars it is connected with. When the application, that processes all the data from all sectors on a highway, asks for this content, the access point looks into its CS and searches for any matches with the incoming Interest. The answer on the Interest message is a single Data message containing all the data of the matching entries.

5.2 Proposed architecture

Out of the four possible architectures described in Section 5.1, we believe that architecture 3 is the least feasible option. Mainly because routing could be quite complex and in the case IP is used to route traffic the whole point of using ICN to create a decentralized network where nodes in between cache and/or aggregate data disappears. Architecture 2 is simple and effective, but since it is a push-based design it is not in line with philosophy of ICN. The two remaining architectures 1 and 4 will fit the most within the philosophy of ICN. Scenario 1 is most in line with the philosophy of ICN since it is the consumer only that sends an Interest message to the producer which answers with a single Data packet. However, it is not very effective since only

one Data packet will be accepted by the access point. Therefore, we think that scenario 4 is more appropriate to use since every car can be pulled by its unique name. Thus all information created by the cars could be used and the routing problem is being solved by letting the car register itself when it connects to an access point.

6 Implementation Details

In this section we will further describe the architecture we chose in Section 5. First we will describe the architecture more in detail, whereafter we describe the proof of concept we made.

6.1 Detailed process

Subscription

When a car connects with a new access point, the car will send an initial Interest message with the unique name of the car and its network address. In this case the unique name of the car is the license plate (but it can be anything as long as it is unique) combined with the network address (e.g. MAC address or IPv4). To send data to the access point we use longest prefix matching (which is only available in NDN). An example of a possible Interest name is shown in Listing 5. The "access point name" is used to route the Interest message to the access point and the "Data" part is where the unique name of the car and the network address is placed. When the access point receives an initial Interest message from a car, it creates a static route based on the information from the initial Interest message. This route is created with an expiration time to ensure that the routing table is not growing until infinity but stays as clean as possible.

```
/nl/tno/A10/s38/sap/udp4:10.0.1.2/7-ABC-44
|                  |                     |
|__________________|_____________________|
 Access point name          Data part
```

Listing 5: Example of Interest name

Data polling

After the the static route is installed the car is added to the polling list. Every x_{sec} the access point will send an Interest message to all cars in the polling list. When the access point receives the data from the car, initially the data will not be processed by the access point but only stored in the CS of the access point. When a car does not answer an Interest message in x_{ms} then the car is considered not available in the current sector, thus the static route and the entry in the polling list are deleted.

Data aggregation

The data that has been polled out of the cars, have a lifetime of x_{sec} to guarantee its freshness. Now data that has been polled is available in the CS of the access point, but the data still needs to be processed and analyzed by an application which can make the data of use for the end user. Now each access point produces an object which represents all data that a specific access point has gathered. The data object the access point produces can easily be propagated using routing protocols since the access point is available in the network for a long period of time.

When the application sends an Interest message for specific sector(s), the Interest gets routed through the backbone to the right access point which produces the data. To answer this Interest, the access point will look into its CS to find all content matching the Interest. Hereafter, the access point will collect the data of all matching content and send a single Data message.

An example of how a Data message could look like is shown in Listing 6. In this example, the application requested the sensor data of sector 38 and 39. The answer contains a dictionary with two keys (one for each sector) and each key contains a value with an array containing the speed of each car in that sector.

```
{
  "s39": {
    "speed": ["165", "127", "125", "122", "120", "165", "122", "129", "128",
    "127"]
  },
  "s38": {
    "speed": ["165", "133", "135", "132", "137", "122", "127", "131", "145"]
  }
}
```

Listing 6: Example of a Data message with aggregated data

6.2 Proof of Concept

We made the proof of concept only for NDN and not for CCN because currently the only way to send data inside an Interest packet is to use longest prefix matching. Since longest prefix matching is not possible in CCN this proof of concept could not be built for CCN. Besides that, there are far more libraries available for NDN (e.g. c++[2], Java[3], JavaScript[4], and Python[5]) then is the case with CCN[6]. To create an NDN application we used the Python module pyndn[5].

The proof of concept consist of three kind of nodes (shown in Figure 6): two car simulators, one access point, and the one application. Because it is hard to use actual cars, we simulated the content production of a car by using a Python script that creates the content a car could create and to make it dynamic, the content changes randomly overtime. Both car simulators are connected over the same LAN to the access point, the same goes for the application which is directly connected though LAN with the access point.

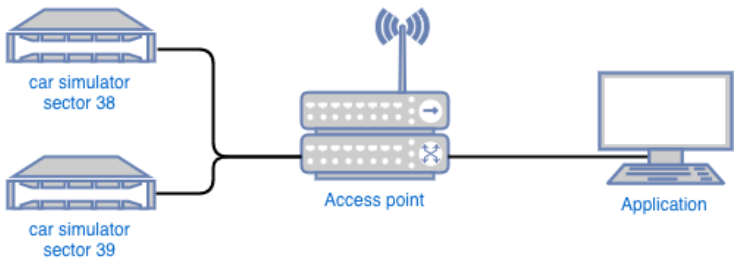


Figure 6: Proof of Concept

When the application receives data from the access point, it will replace the old data of each sector. In this way, old information is still available when there are no cars present in the sector.

[2]https://github.com/named-data/ndn-cxx
[3]https://github.com/named-data/jndn
[4]https://github.com/named-data/ndn-js
[5]https://github.com/named-data/PyNDN2
[6]https://github.com/PARC/CCNx_Distillery

For example, the temperature of two hours ago could still be relevant information. Each time new information of the sector is placed, the time stamp gets replaced, in this way the application can calculate how old the data actually is. Listing 7 shows the data structure we use to save information within the application.

```
{
   "s38":{
      "speed":127.0,
      "timestamp":1466668075.271727
   },
   "s39":{
      "speed":134.0,
      "timestamp":1466668131.74673
   }
}
```

Listing 7: Data structure of information within the application

Listing 8 shows the output of the application itself, where the timestamp is compared with the current time. One can see that of sector 38 is fresh and last data received from sector 39 is more then 21 hours ago.

```
Send : /nl/tno/A10/s38s39/sap
Sector: s39  average speed: 132.0  age of data: 78399.24 seconds
Sector: s38  average speed: 134.0  age of data: 0.0 seconds
```

Listing 8: Output of application

7 Conclusion

In this paper we looked at how Content Centric Networking compares to Named Data Networking with regards to IoT applications. We found that for the most part CCN and NDN are fundamentally very similar. However, there are differences how these fundamental principles come to life in each project. For example, the packet format differs between both implementations as CCN describes fixed-size headers whereas NDN uses variable-size headers. Also the way forwarding works differs, for example, when a CCN node receives an Interest packet it will first look into the CS and then create a PIT entry and with NDN this is the other way around. The CCN name must match exactly while the NDN naming scheme does not have to match exact making it possible to use more labels then necessary. For our use case, we found that NDN is more appropriate then CCN. Mainly because NDN does not need to have exact name matching, which makes it possible for us to send data upstream which makes the routing possible for fast converging IoT devices.

8 Future Work

The philosophy of ICN is to be a pull-based architecture where the communication is driven by the receiving end, e.g. the data consumer. To receive data, the consumer must send an Interest message to the producer of the data, on its turn, answers with the appropriate Data packets. Mainly because in our use case nodes can change network connections in rather fast manner, which makes participating in a traditional routing protocol difficult. To solve this problem, nodes need to register itself to an access point. But since there is no field in both CCN and NDN Interest messages to do so, pushing data could be a problem. The support of NDN for using more labels than needed, can be a solution to this problem, although it is not meant for that. We suggest that there should be some field inside the Interest packet to send data to the producer. By enabling data to be sent inside an Interest packet, it should be easier for an application developer to create software. Although we think it is necessary to research the possibilities of using data inside Interest packets. One of the conflicts that could arise with sending data inside Interest packets is the fact that they could get aggregated. This makes sending data through Interest packets unreliable. However, when not having possibilities to push data, one can choose to use IP-based application instead of ICN because ICN in this sense is not flexible enough.

For our use case, the different forwarding schemes used by CCN and NDN do not really make a big difference. However, there could be differences for IoT devices with low processing power and/or low energy capacity. With these constrained devices, it could make a difference on how forwarding works, in terms of how efficient it is. For example, NDN generates for each Interest a 4-octet long byte-string nonce to uniquely identify Interests and CCN does not. Also the question which way is faster for constrained devices, looking in the PIT first or in the CS, and which one uses cost less energy?

Acknowledgements

We would like to thank Bastiaan Wissingh for his support during this project and for proofreading our proposal and final report. Furthermore, we would to like to thank Lucia D'Acunto for for guiding us throughout the course of this project. Lastly we thank Ray van Brandenburg for the support in the first week and proofreading our proposal.

Appendices

A Forwarding flow-charts

A.1 CCN Interest forwarding

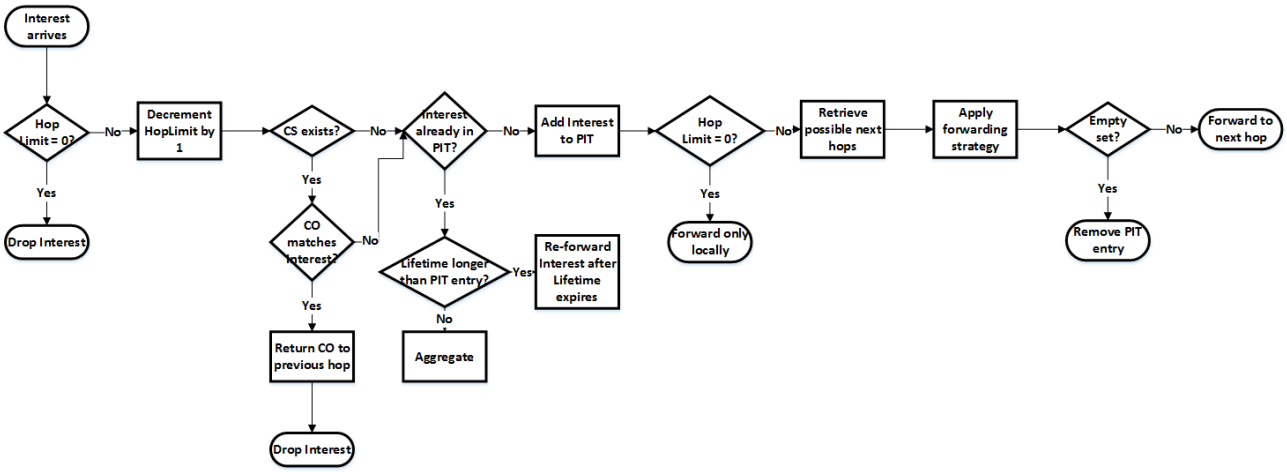


Figure 7: CCN Interest Forwarding

A.2 CCN Content Object forwarding

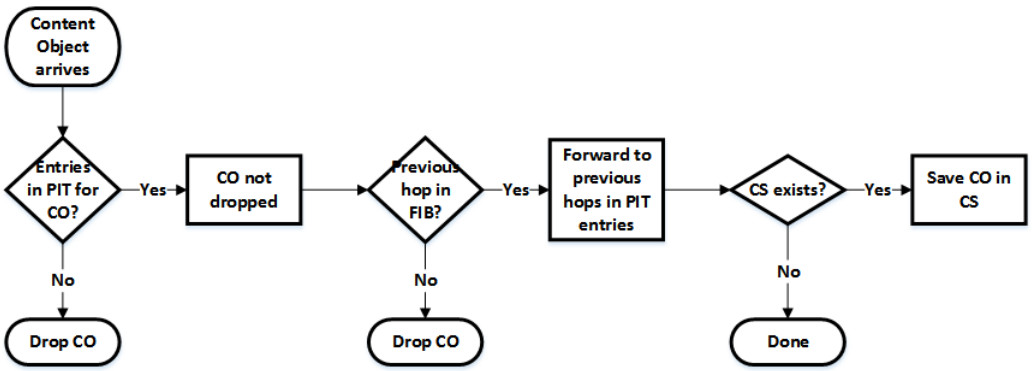


Figure 8: CCN Content Object Forwarding

A.3 NDN Interest forwarding

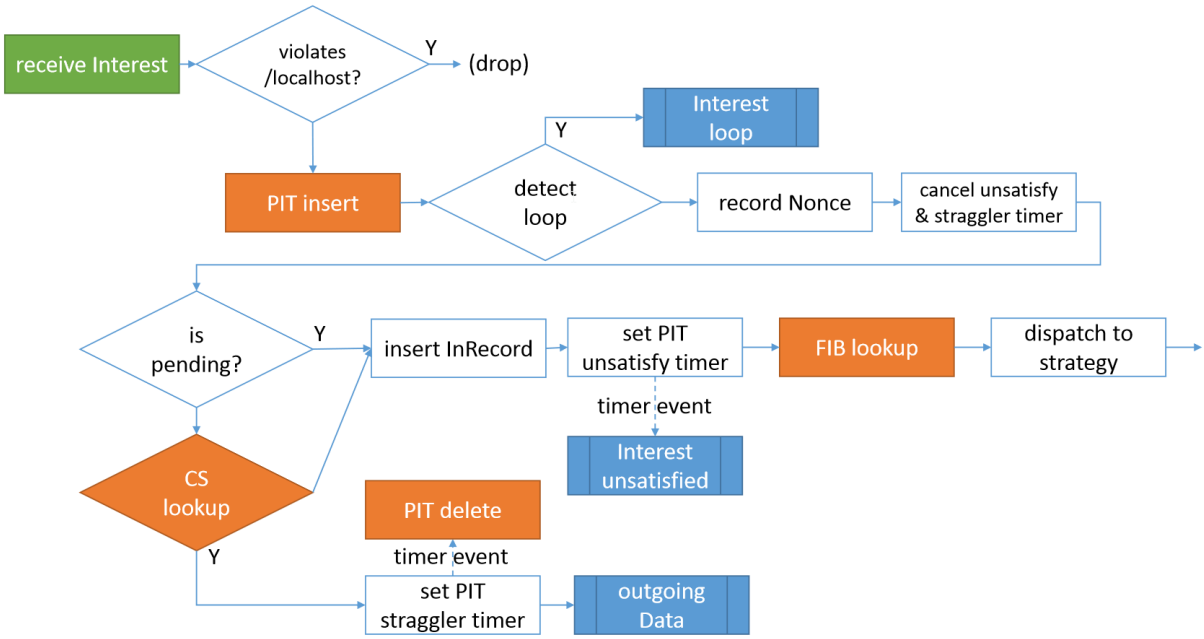


Figure 9: NDN Interest Forwarding

Source: NDN Developer Guide [1]

A.4 NDN Content Object forwarding

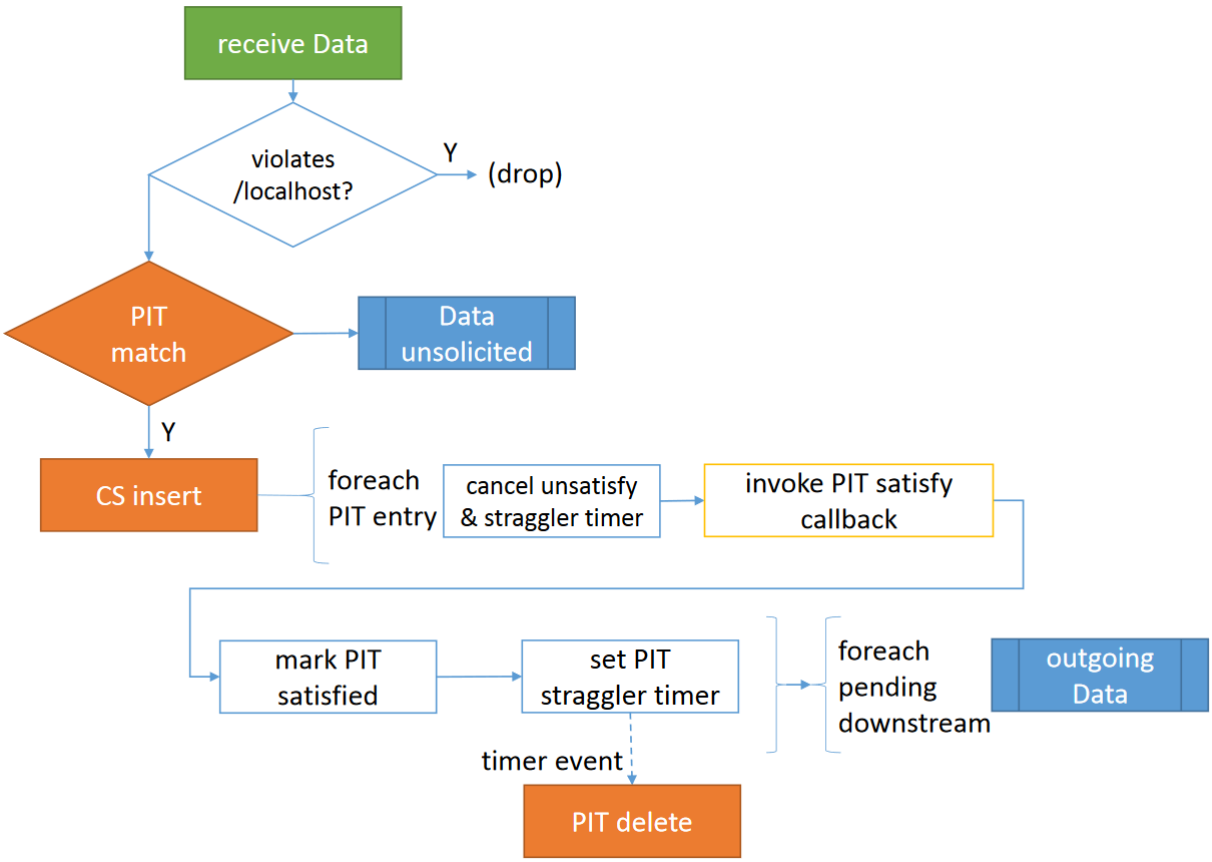


Figure 10: NDN Content Object Forwarding

Source: NDN Developer Guide [1]

References

[1] Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto, et al. Nfd developer's guide. Technical report, Technical Report NDN-0021, NDN, 2016.

[2] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. Scaling ndn routing: Old tale, new design. Technical report, NDN, Technical Report NDN-0004, July 2013.[Online]. Available: http://named-data. net/techreports. html, 2013.

[3] Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C Schmidt, and Matthias Wählisch. Information centric networking in the iot: Experiments with ndn in the wild. *arXiv preprint arXiv:1406.6608*, 2014.

[4] "N.G.J.C. Bukkems and E. Folles". Alternatieven voor lusdetectie. `http://docplayer.nl/2659600-Alternatieven-voor-lusdetectie.html`. Accessed: 2016-06-13.

[5] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, Ion Stoica, and Scott Shenker. Rofl: Routing on flat labels. *SIGCOMM Comput. Commun. Rev.*, 36(4):363–374, August 2006.

[6] Jose J Garcia-Luna-Aceves. Name-based content routing in information centric networks using distance information. In *Proceedings of the 1st international conference on Information-centric networking*, pages 7–16. ACM, 2014.

[7] "ICNRG". Background. `https://irtf.org/icnrg`. Accessed: 2016-06-01.

[8] Dirk Kutscher, Suyong Eum, Kostas Pentikousis, Ioannis Psaras, Daniel Corujo, Damien Saucez, Thomas C. Schmidt, and Matthias Waehlisch. ICN Research Challenges. Internet-Draft draft-irtf-icnrg-challenges-06, Internet Engineering Task Force, March 2016. Work in Progress.

[9] Vince Lehman, AKM Mahmudul Hoque, Yingdi Yu, Lan Wang, Beichuan Zhang, and Lixia Zhang. A secure link state routing protocol for ndn.

[10] Anders Lindgren, Fehmi Ben Abdesslem, Bengt Ahlgren, Olov Schel, Adeel Mohammad Malik, et al. Design choices for the iot in information-centric networks. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 882–888. IEEE, 2016.

[11] Priya Mahadevan "Marc Mosko, Ignacio Solis and Ersin Uzun". Ccnx 1.0 naming: Transforming network addresses to application value, 2014.

[12] marc.mosko@parc.com. CCNx Messages in TLV Format. Internet-Draft draft-irtf-icnrg-ccnxmessages-02, Internet Engineering Task Force, April 2016. Work in Progress.

[13] marc.mosko@parc.com, Ignacio Solis, and Christopher Wood. CCNx Semantics. Internet-Draft draft-irtf-icnrg-ccnxsemantics-02, Internet Engineering Task Force, April 2016. Work in Progress.

[14] Marc Mosko. Ccnx 1.0 protocol introduction. Technical report, Technical report, Palo Alto Research Center, 2014.

[15] Marc Mosko. Ccnx 1.0 routing in ccn. Technical report, Technical report, Palo Alto Research Center, 2014.

[16] "NSF". How does ndn differ from content-centric networking (ccn)? http://named-data.net/project/faq/#How_does_NDN_differ_from_Content-Centric_Networking_CCN. Accessed: 2016-06-01.

[17] "NSF". Named data networking: Motivation & details. http://named-data.net/project/archoverview/. Accessed: 2016-06-06.

[18] Zhong Ren, Mohamed Ahmed Hail, and Horst Hellbrück. Ccn-wsn - a lightweight, flexible content-centric networking protocol for wireless sensor networks. In *ISSNIP*, 2013.

[19] Divya Saxena, Vaskar Raychoudhury, Neeraj Suri, Christian Becker, and Jiannong Cao. Named data networking: a survey. *Computer Science Review*, 19:15–55, 2016.

[20] Junxiao Shi. Nfd: Dead nonce list. https://github.com/named-data/NFD/blob/master/daemon/table/dead-nonce-list.hpp.

[21] Ignacio Solis. Ietf 90 - july 2014: Ccnx 1.0 changes from 0.x. https://www.ietf.org/proceedings/90/slides/slides-90-icnrg-10.pdf.

[22] V Vetriselvi, Shaik Fayaz, Arun Balaji, T Velu, et al. Recognition and interception of hand gesture that avoids keyboard strokes. *Transylvanian Review*, 24(6), 2016.

[23] "Greg White and Greg Rutz". Content delivery with content centric networking, 2016.

[24] George Xylomenos, Christopher N Ververidis, Vasilios A Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V Katsaros, and George C Polyzos. A survey of information-centric networking research. *Communications Surveys & Tutorials, IEEE*, 16(2):1024–1049, 2014.

[25] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobssen, et al. Named data networking. http://www.sigcomm.org/sites/default/files/ccr/papers/2014/July/0000000-0000010.pdf. Accessed: 2016-06-15.

[26] Yanyong Zhang, Dipankar Raychadhuri, Luigi Alfredo Grieco, Emmanuel Baccelli, Jeff Burke, Ravi Ravindran, Guoqiang Wang, Bengt Ahlgren, and Olov Schelen. Requirements and Challenges for IoT over ICN. Internet-Draft draft-zhang-icnrg-icniot-requirements-01, Internet Engineering Task Force, April 2016. Work in Progress.