# Evaluating the use of ICN for Internet of things

Johan Carlquist

October 18, 2017

**Abstract**

Your abstract.

# Contents

# 1 Introduction

The main paradigm of current networks today is called *host centric* and most of our communication is defined as end-to-end between these hosts. It is predicted that in 2020 we reach around 50 billion Internet of Things (IoT) devices [1] and the usage of these devices often imply information centric usage patterns [2].

Information-centric networking (ICN) is a communication paradigm for the future Internet that is based on *named data* instead of *named hosts*. The communication is defined within requesting and providing named data, decoupling senders from receivers. This could make it possible to integrate storage caching within the network infrastructure [2] which could lead to improvements in the network as a whole.

- Rewrite.
- Write this lastly.

## 1.1 Motivation

- Remove this and put it under introduction instead?

## 1.2 Problem statement

The purpose of this thesis project is to evaluate the performance and feasibility of using ICN in the IoT-domain. The scenarios are normal usage patterns between devices including producing and consuming large set of data created in a periodic interval. By evaluating the performance of ICN components and the application as a whole, one can provide a comparision between such an application and a network reference point. Ideally it should be possible to divide and present where most of the time is spent operating under ICN, therefore a part of the goal of this thesis project is to provide an in depth analysis of where most time is spent when using the ICN application.

Another part of the performance goal is to investigate how well ICN can scale as the network increases in size, main focus is considered when several consumers want to access data that is provided by one producer.

ICN is by nature a pull paradigm where a consumer has to initiate a request of a particular data in order to retrive it. This is in constrast to the several IoT systems where publish subscribe approach is in greater use, such a system pushes the data to the user when it is produced. By evaluating the feasibility, a major goal is to investigate if it is possible to achive similar outcome with ICN together with a onetime subscription model and if such a system would be stable. With the onetime subscription model, a consumer tries to pull the data from the producers by initiating a request just before the data has been created.

## 1.3 Delimitations

Even though there are several different ICN approaches and flavors as Psirp, Netinf among others, there will no comparision between them in this thesis. CCN will be the only ICN implementation regarded in this thesis due to the fact that it has a stable version running. Furthermore, this thesis will not develop any further functionallity on the CCN implementation for Contiki-OS. The current implementation is to be considered good enough during the time for the thesis. Exceptions is made

for functionallity regarding monitoring and measurement metrics that will have effect on the evaluation. Different cache strategies for the local storage at a CCN node and other functionallities that could be nice to have, but not necessary, and it is to be considered out of the scope of this thesis. Furthermore, no aspect of power consumption or measurement will be taken into considered in this project.

CCN will be the only ICN implementation covered in this thesis. Alternative implementations such as Psirp, Netinf among others, will not be covered at all due. Furthermore, this thesis will not develop any further functionallity on the current CCN implementation for Contiki-OS. The current implementaion is to be considered sufficient. Exceptions are made for functionallity regarding monitoring and measurement metcis that will have effect on the evaluation. Different cache strategies for the local storage at a CCN node and other functionallities that would be desirable, but not necessary, is considered out of the scope of this thesis. Furthermore, no aspect of power consumption will be taken into account for the thesis.

## 1.4 Research methodology

Currently there is a CCN-lite implementation of the IoT light-weighted Contiki OS. In this project, a qualitative and quantitative performance evaluation will be carried out through experimentation with this implementation. The hardware representing the sensors is going to be the Texas Instruments sensortag CC2650 which will run the CCN-lite implementation. Tests regarding scalability will be conducted using emulated sensors.

Since the thesis is based on experiments a lot of measurements will be conducted. The methodology used will be short cycles and small loops between each measurement. This iterative approach has the advantage that it can give answers to questions that are not specifically asked and it can also be used to see changes over time. In order to make the testing intervalls short and fesable, a lot of effort has been conducted to create smart scripts that automate testing and representation of the data so it becomes visual to the tester.

Different software tools will be programmed to measure the performance. In order to evaluate the performance, various types of measuring tools will be conducted. These tools has either a general or a specific purpose. A general purpose tool can for instance collect the measured data from the test, and make it representable. Whereas specific purpose tools can be ones that measure the processing time on a sensor or measure the latency time.

## 1.5 Expected contributions

This paper contribute an experimental evaluation, with performance and feasibility aspects in focus, of using CCN on industry hardware in a typical IoT enviroment.

There are several other expected contributions of this thesis project, excluding the paper. One is improving the current CCN-lite application for Contiki-OS, mostly to make it more stable and reliable when runned on industry hardware. Another contribution is the development of latency measurement software, especially for IPv6, for the CCN-lite project. Moreover, software enabling a consumer to retrieve data from the publisher through a one-time subscription model described in prior parts.

Furthermore, several testing softwares have been developed to make this thesis

possible. Although a lot of time and effort has been conducted into these software implementations, they will not be covered in this report.

## 1.6 Related work

When Jacobson et al. released their paper *Networking named content* in 2009 it sparked ideas of an alternative approach of communicating in contrast to current IP networking[3]. They implemented a prototype which replaced the IP with CCN in the network stack and proved that it could be a potential alternative for the future Internet. Since then, several research papers have been published comparing different ICN alternatives and their potential benefits and trade-offs when implemented as a network service[2]. Studies prove that it could be a suitable replacement of current IP networking structures, but there is a need for more performance analysis and studies[2][4].

However, it was not until the last couple of years that the research community started to investigate the feasibility and applicability of using ICN in the IoT domain. Several studies, the majority being literature studies and theories, have been conducted recently or are currently ongoing. There only seem to be two implementation studies available to date.

In a litterature study conducted by Ahlgren et al. [5], they conclude that an advantage with ICN, the naming of data, is independent of the device that produces it. The decoupling between a publisher and subscriber of data could improve performance in loosy networks. Challenges reside in the naming of data that is produced periodically over time where a major issue is to retrieve the *latest* value in that sequence. Potential solutions could be to implement a *one-time subscription*, where the request is stored in the cache at the node until the data becomes available[5].

Another litterature study provided by [6] argues ICN is by nature close to the IoT domain. They also conclude that there is a need of further investigations regarding if ICN should be implemented as an overlay of existing IP infrastructure, coexist with IP, or if it should be a replacement in the same manner as proposed in [3].

A paper from INRIA was the first major academic project which ported CCN-lite to the IoT operating system RIOT[7][8]. That project was the first trial of implementing ICN without any IP protocol. They compared their CCN-lite implemententation and a regular 6LowPAN/IPv6/RPL approach and saw that there were several advantages using ICN over IP. Although they identified several areas where further work needs to be done, they argue that ICN is applicable in the IoT domain.

Another implementation of CCN for IoT devices was a thesis project conducted by Yanqui Wu at SICS/KTH[9]. He ported the CCN-lite functionallity into another IoT operating system, Contiki-OS, and lay the software in the application layer instead of the network layer as INRIA. Although some evaluation was done, there was no further investigation on how well CCN performs at the application layer.

## 1.7 Structure of the Report

This report is structured into [#][five] sections, omitting the introduction.

- This is under construction.
- Background-section
  Section 2 will discuss background knowledge regarding the problem to this date. Technical details is presented that enable the reader to understand the details

for the rest of the report. The background will first cover the network stack focusing on IoT devices, thereafter ICN will be convered generally and CCN in more depth. A breif overview of the Contiki-OS will gain knowledge about the OS running on the hardware motes. Lastley a short summary of the hardware used in this project.

- Setup-section, Based on the background, the setup used in this project will be outlined.
- Metod-avsnitt, can not decide if this should be an own paragraph or inside under the result. TBD.
  In order to fully understand the background, experiments, results and discussions, Section 3 will serve as an outline of the testing.
- Result-section
- Discussion-section
- conclusion/futurework-section

# 2 Background

Describe the greater context, what are the technologies and protocols figuring in this thesis. In beginning mention a little bit of both IoT and ICN sort of waving them together.

## 2.1 Internet of Things - network stack

- Write about growing number of devices
- how the devices come into play and where they're used.
- Describe the devices similarity, that they often are the same. High level of hetrogenous.

Write about the growing nubmer if devices. How they're coming into play and where they're used.

### 2.1.1 IEEE 802.15.4

The IEEE 802.15.4 standard intends to offer the fundamental lower network layers for wireless personal area networks (WPAN). The standard focus is on providing a low cost, low power consumption and low data rates between inexpensive wireless devices. The standard only provides the MAC and PHY layers, leaving the upper layers to be chosen by the applicant[10] [11]. Due to the special PHY layer and to keep the transmission times short and resistant against failures, it does not exchange standard Ethernet frames with maximum transmission unit (MTU) of 1500 octets. The MTU of 802.15.4 is instead set to 127 octets. The communication range is set up to 10 meter and a maximum data transfer rate limited at 250kbit/s. Depending on wireless technology and how constrained the device is, the maximum transmission speed can be set to as low as 20 kbit/s. [<rewrite this>] It is important to stress that the 802.15.4 standard does not compete with the regular 802.11 standard, where costs are not as critical and security and speed are more demanded. Today transfer rates up to gigabites is possible with the latest 802.11 standard, these speeds are hardly unneccesary in the IoT domain.[<rewrite this>]

There are two different types of network nodes that can exist in a 802.15.4 network[11]. Full functional device(FFD) and reduced function device(RFD). A FFD node implements all communication funcionallity the 802.15.4 standard offer, it can communicate with any other device in the network. A FFD node may therefore also route data from other nodes. When doing that the node is also called a cooardinator. If all communication in the network is routed through a dedicated FFD node, it is called a PAN coardinatior. A RFD has a reduced level of functionallity and is meant to be extremly simple. Such devices are always an end node in a network and can only communicate through or with a FFD. They can never act as a cooardinator due to their limited capabilities.

The two main network topology forms that are used within the 802.15.4 standard, are the star topology and the peer-to-peer topogoly, shown in figure 1. In star topology, all devices are required to only communicate to a single central device called the PAN controller. An advantage with this topology is that it makes it easy to manage and support. The drawbacks of using a star topology structure are bigger, for instance will it limit the area that can be covered geographically since all data has to be routed

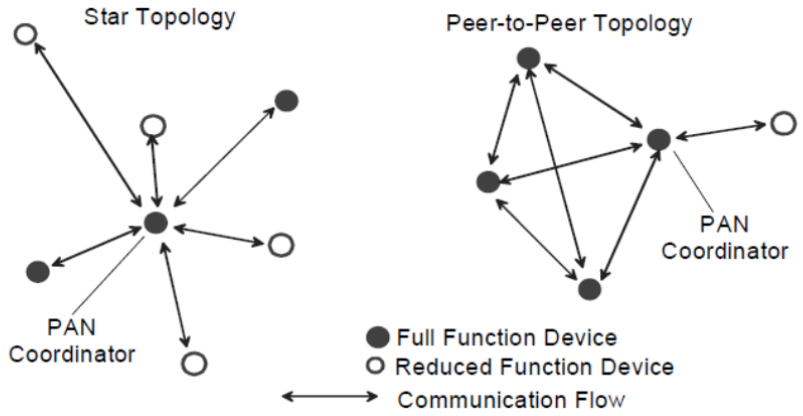


*Figure 1: Topology structures in 802.15.4 networks. Star structure on the left, Peer-to-peer on the right.*

through one device and the distance a node can cover is set to be at a maximum of ten meters.
The peer-to-peer topology can have an arbitrary number of connections to each other within the network. Devices can communicate with each other, not only through the PAN controller, with exception for communication between RFDs. There are several advantages by using the peer-to-peer structure, for instance since devices can route traffic via other FFD devices, the network coverage can be easily increased.

exisiting usage of 802.15.4... maybe a subsection of Zigbee too?

### 2.1.2 IPv4, IPv6

Since the introduction of Internet Protocol version 4 (IPv4) in 1981[12], it has been the backbone of the Internet and as the network layer in the OSI model. The protocol defines an address space of 32 bits and the total number of unique addresses that is available with IPv4 is around 4 billion. Today, the IPv4 address space is exhausting at a rapid speed and there is not enough addresses left to handle the increased number of devices that will be connected to the Internet in the future[Make citation!!].

In response to the shortage of address space, among other things, the Internet Protocol version 6 (IPv6) was formulised and defined as a successor to IPv4 in 1998[13]. The IPv6 protocol defines the length of an address of 128 bits, which lead to a total address space of $2^{128}$ equal to $3.4 * 10^{38}$ unique addresses. With an address space of this size, it will be sufficient for all IoT and Internet devices to have an own IP address. IPv6 requires that every link to the Internet has an MTU of 1280 octets or greater. In case this need can not be met, fragmentation and reassembly must be provided at layer below IPv6[13].

### 2.1.3 6LowPAN

At first glance, it may seem straightforward to send IPv6 data packets on a 802.15.4 network. However, there are incompabilities between the two formats making it hard for them to cooperate. For instance, the largest frame size of 802.15.4 (127 octets)

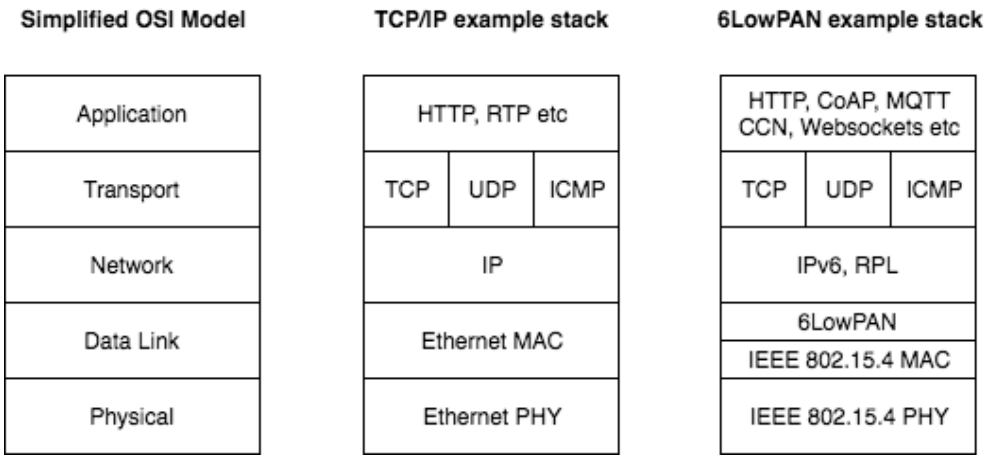


*Figure 2: The simplified OSI model, an example TCP/IP stack and an illustration of an example 6LowPAN protocol stacks.*

is considerably less than the required MTU of IPv6 (1280 octets) [14]. Futhermore, the IPv6 header is 40 octets long which is almost a third of the total 802.15.4 MTU (at least 25 octets). This leaves only 62 octets for upper-layer protocols as UDP or ICMP. That makes it impossible in the first case and infeasable in the second, to build IPv6 direclty on top of the 802.15.4 MAC layer as in a regular IP protocol on the Ethernet MAC layer in the IP stack, shown in figure 2.

To address these issues, among several, "IPv6 over Low-Power Wireless Personal Area Networks" (6LoWPAN) was established in 2007 as an adoption layer between the IEEE 802.15.4 MAC-layer and IPv6. 6LowPAN makes it possible to transfer IPv6 packets over a 802.15.4 network through fragmentation and reassembly, and IPv6- and UDP header compressions to shrink the packet size. Through header compression strategies, it is possible to shrink down the IPv6- and UDP header, toward as little as 4 octets in total (instead of 48 octets) [14]. Other features of 6LowPAN is the neighbor discovery and mesh routing support. Even though there is no limitation to only use UDP, for simplicity and performance reasons it is more favorable to use UDP over TCP as the transport protocol with 6LowPAN.

### 2.1.4 MQTT, MQTT-SN

Message Queuing Telemetry Transport(MQTT) is a open lightweight publish-subscribe messageing protocol, designed for constrained devices with low-bandwidth and/or unreliable networks targeting Machine-to-Machine (M2M) communication[15]. The protocol reside in the application layer in the OSI model, assuming that the underlaying network structure provides a point-to-point, session-oriented data transport provided by example TCP/IP [16]. This assumption makes the protocol unsuitable for devices that can not hold their own TCP/IP stack, which lead to MQTT-SN described further down. The publish-subscribe message pattern require a message broker, which is responsible for distributing messages to the interested clients.

The publish-subscribe pattern can be described by a server, or sensor, acting as a publisher/producer of information and a client as the consumer/subscriber of information. A client subscribes on a specific topic, set of data, that resides on the

server/sensor. When the server has produced data for the specific topic, it will send that information towards the client. The information is going through a broker that handles all the information regarding which devices subscribe to which publisher. The broker is usually located in a traditional network due to its higher performance regarding bandwith and processing capabilities.

MQTT-SN, where the extension stands for sensor network, is a MQTT version that is adapted for wireless communication. It is optimized to be implemented on low-cost, battery-operated devices with limited or constrained storage and processing capabilities[17], perticular targeting IoT and sensor devices.
Where MQTT uses string characters as topic names, MQTT-SN uses numeric IDs which reduce the size of the packets in favor of readability. Furthermore, MQTT-SN, in contrast to MQTT, does not depend on a connection-oriented transport service (TCP/IP), it is able to work with other transport protocols such as UDP/IP, ZigBee or others.

[more about brokers and its role.]

### 2.1.5 CoAP

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol to be used with constrained devices and networks, and between M2M communication[18]. CoAP resides in the application layer, above transport layer, in the OSI model stack. It provides a client/server interaction model between application endpoints similar to the HTTP standard. CoAP is based on the REST architecture and follows the general design to manipulate data in a request/response manner. The methods GET, POST, PUT and DELETE are similar to HTTP, but not identical.

While HTTP uses TCP as transportation protocol, CoAP data is sent asynchronously over a datagram-oriented protocol such as UDP. Due to the implementation of UDP, features like resend lost datapackets, and acknowledge-messages are missing in the transport layer[ref kid]. This functionallity has in some extent been moved into the CoAP protocol and is called Messages.
CoAP defines four different type of messages: confirmable, non-confirmable, acknowledgment, reset. They occupy 2 bit out of 32 bit of the total CoAP header. A confirmable message provides reliability by retransmitting a message until a recipient sends an Acknowledgment message with the same Message ID back to the requester. If a requested message can not be handled by the recipient, a reset message will replied instead. When a message does not require reliable transmission (no acknowledgment is needed), a non-confirmable message is sent. These non-confirmable messages will still have a message ID in order to detect duplication.

[note, maybe reorder so first coap, then request/response, then Messages? Add about in network caching? Also add source to second paragraph]

## 2.2 Information Centric Networking

Information-Centric Networking (ICN) is a communication paradigm for the future Internet that is based on *named data* instead of *named hosts*. It represents an evolution of the Internet from todays host-centric networking, in particular ip networking, towards an information-centric approach.

There are several different approaches of implementing the ICN paradigm, but there are fundamental ideas that they all follow regardless which implementation is used.

In this section will continue describing the gerneral ideas. However, the thesis overall will only consider the *Content Centric networking* approach described more in depth at section [ref till CCN].

Some of the main features in the ICN architecture are the possibility of in-network storage for caching data, multiparty communication through replication, decoupling senders and receivers, and that data is named[2]. In ICN networks, an information request may not only be satisfyed by locating the original information source, it is possible for any of the in-network caches to reply the request with the desired data if they hold any copies of it. After a request is sent from a user, it is the responsibility of the network to locate the best source that can provide the information. Due to the fact that information/data is named, addressed and matched independently from its location, the data may be located anywhere in the network[icn research][18][19]. Hence the argument that the ICN approach decouple the information from its source.

### 2.2.1 Content-Centric Networking

- motivation, one type of ICN architecture.

The CCN communication is driven by consumers of data. There are two types of packets in CCN, *Interest* and *Data*. A consumer issues an *Interest* message to request an information object on the network. Any node that recieved the interest and containing the requested information, will respond by sending the *Data* back to the *Interest* issuer. A *Data* packet is only sent in response to an *Interest* message, upon response, the interest message will cease to exist in the network.

A key feature of CCN is that the content names are hierarchical. This allows name resolution and routing information to be aggregate which in turn is critical in order to scale the network. The content name can be similar to the way we access URLs, for example a valid content name could be */sics.se/kista/floor/six/sensor/two/temperature*. However, there is neither a strict need for them to be human-readable nor tobe a URL. The prefix */sics.se/kista/floor/six/sensor/two/* could easily be exchanged to become either a hash value or just an integer, say *2* (representing the sensor two), whereby the same data could be accessable by the name */2/temperature*.

It is to be considered an interest hit when any part of the interest name equals the named prefix of the data, for example is it possible that */2/temperature* could be match by */2/temperature/sequence_1*. If the data is produced periodically with sequence numbering, a consumer can 'follow' the data by the same manner once it has a starting point. Another advantage with periodically and sequencing, is that it provide the possibility to ask for data that has not yet been produced. A consumer could potentially issue an *interest* request a short time before the *data* has been produced, which would firstly get the data directly to the user and secondly minimise the latency on the network[ref to bengt2](No performance evaluation of this has been done to date, the thesis will try to answer the feasability of this.).

Even though there are a lot of similiarities to regular routers IP, there are a lot of differences between a CCN router and a regular IP router. Every Content Router (CR) maintain three main data structures: The Forwarding Information Base (FIB), the Pending Interest Table (PIT) and the Content Store (CS).

The FIB is used to map information to on which output interface a Interest message should be forwarded to in order to reach its content. It is very similar to a regular FIB in a IP router, with the major difference that the CCN FIB allows lists of outgoing interfaces instead if just a single entry per object.

The PIT maintain a table for all incoming *interests* request recieved by the CR, their current state and a mapping to which face they came from[Bengt]. When the data packet for a particular *interest* arrives to a CR, the data packet will be forwarded back on a reverse path, towards the face that exist in the corresponding PIT entry. After the data packet has been forwarded towards the requester, its entry in the PIT will be erased. Whenever an entry is dropped or lost, for instance due to timeouts, it is up to the consumer to issue a new interest request[?].

The CS act as a local cache for information objects that has passed through the CR.

With the use of the CRs, CCN has great support for data caching. As stated earlier, once a *interest* is receievd, the CR will look through its CS in order to find matching data. Once the data is on the reverse path to the consumer, it will be put in the CS for an limited period of time for futher use. Although there is several benefits using the cache in a distributed network system, it should be pointed out that this is not a long-term storage since a router can not hold infinite number of data. Nor is it useful for data object that is requested at most once, since the benefits only occur when the data is requested a second time [5][2].

An example of CCN in action is illustraded in figure 3. Here, a subscriber wants to retrive the indoor temperature data from the producer. Subscriber1 sends an *interest* for data */temp/indoor* towards CR1. When it arrives, CR1 looks for data in its CS that matches the requested prefix of the interest. Since there is no match in the CS, the router performs a lookup on the longest prefix that matches its FIB in order to decide where to forward the incoming interest. When the match in the FIB is found, the router inserts the interest, with the incoming interface, into the PIT.
The same procedure happens for CR2 and the interest will be put in the PIT and forwarded to the producer. When the *interest* reaches the producer, it matches the name of the *data* and thereby the *interest* message is discarded and the *data* is sent back towards CR2. When the data is received, CR2 stores the data in the cache. Thereafter it performs a longest-prefix match in its PIT to get which interface it should respond to. In this case, it will respond to CR1 and the same forward back procedure will occur at CR1 until the data reaches the subscriber1.
When subscriber2 later on wants the same content, he sends an *interest* to CR1 and will retreive the data directly from its cache and thereby reducing the network traffic.

### 2.2.2 Using ICN for IoT

The usage of IoT devices most often implies an information centric pattern. In many scenarios, the main goal are the data and services and it is less important to communicate with a specific device [5]. Where users and/or devices rather consume content, generated by an IoT device, through the network than connecting directly to a specific device or host. Therefore one could argue that naming the data is more important than naming the devices.
Depending on topology structure of the IoT network, the caching mechanisms ICN provides could help constrained IoT devices to avoid unnecessary transmissions when distributing its data into multiple places. Storing cached data in the network could

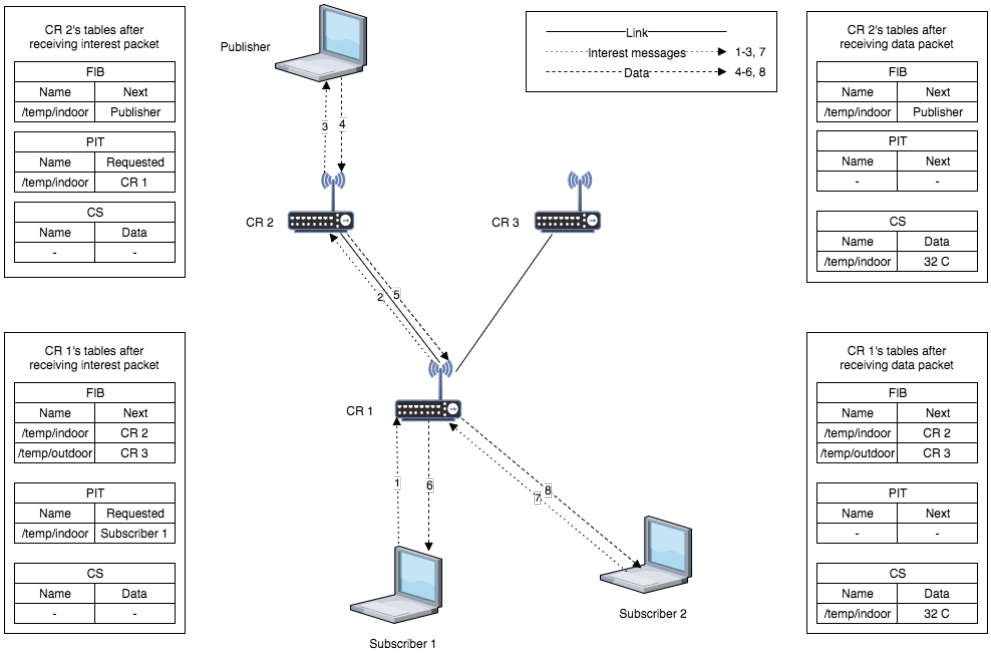


*Figure 3: The CCN architecture builed up by content routers (CR), forwarding information base (FIB), pending interest table (PIT) and content store (CS), inspired by [ref survey ICN]*

also potentially save battery and network bandwidth of an IoT device.

There are several reasons why it is better to use an ICN approach in the IoT domain. Everything always depends on application and usage, but there are some pros and cons using ICN in the IoT domain.

- + naming of data
- + distributed caching
- + Decoupling between publisher and consumer
- + Sending request for the future.
- advantages of using ICN in IoT domain.
- disadvantages Even though the major advantage of using named data in IoT, it comes with some concerns. How to create and format efficient names suitable for huge numbers.

### 2.2.3 CCN-lite

CCN-lite is a lightweighted, functionally interoperable implementation of CCN [19]. There are several platforms that are supported such as UNIX, Linux, Android, Arduino and several other. It uses a small code base of C, less than 2000 lines of code, without any compromise of the CCN functionallity. CCN-lite runs over UDP and Ethernet, and support packet fragmentation.
A previous thesis worker at SICS implemented a version of CCN-lite to the Contiki-OS platform.

## 2.3 Contiki-OS

Contiki OS is an open source operating system that is suitable for network-connected, memory-constrained devices that focusing on Internet of Things [20]. It focus on wireless technologies and implement number of IoT protocols including 6LoWPAN, IEEE 802.15.4, RPL, CoAP and MQTT. Contiki provides a full IP network stack with protocols as IPv4, IPv6, TCP, UDP and HTTP. It is designed to operate with extrem low-power systems.

### 2.3.1 6LBR

In order to connect a device to the Internet one can use the 6LowPAN Border Router, 6LBR[21]. It is implemented on Contiki-OS and provide interconnection between IP and 6LowPAN networks. The router assumes an Ethernet interface on the IP side and 802.15.4 on the sensor side. Devices connected to the 802.15.4-to-Ethernet gateway can reach and be reachable to/from the Internet. There is no native support for IPv4, altough there are mechanisms to achive some IPv4 funcitonallity through a NAT64. [is this contributing any to the thesis? Or should it be more focus on Sparrow.]

### 2.3.2 Sparrow

Sparrow is a commnunication format that encapsulates different types of payload on top of IPv6/UDP [22]. The Sparrow border router is based on the original Contiki border router but it has been improved with additional features. It acts as the RPL root and handles all the routing towards the sensors and maintains the network as a whole. The software makes it possible to initiate and hold communications with the remote sensortags. The border router connects the sensor network to the local host (Linux/OS-X or other), making it possible for the applications on the host to reach nodes in the sensor network.

### 2.3.3 CCN-lite portation

A former thesis worker at SICS implemented and ported a version of the CCN-lite[19] to the Contiki-OS platform in 2016[9]. The software is to be considered laying in the application layer and will not replace any other layers in the OSI model. It handles all necessary functionallity that the regular CCN-lite application provide. Depending on how much memory that is available at the hardware, one can tune in how many entries that the PIT-table and the content store can hold.

## 2.4 Hardware

### 2.4.1 Texas Instrument CC2650 SensorTag

The CC2650 sensortag is a wireless microcontroller developed by Texas Instruments [23]. The device is low cost, ultralow power device using the 2.4 Ghz radiofrequency to communicate with technologies such as 6LowPAN, Bluetooth and Zigbee. Due to its ultra low power consumption, the sensor can be powered by battery. The CC2650 device contains a 32-bit ARM Cortex-M3 processor running at 48 Mhz, accompanied by 8 KB of cache and 20 KB of SRAM. It contains a total of 128 KB of programable flash memory, which can be used for different application system such as the Contiki-OS system. The sensor controller supports the measurement of different types of sensor data such as temperature readings, optical light values and more.

### 2.4.2 Zolertia Firefly Slip radio

The Firefly radio slip is developed by Zolertia [24]. The radio slip provides a network infrastructure for the IoT devices, enabling them to communicate efficiently through the air. The Firefly has great routing capabilities due to its support for several communication technologies, among them IEEE 802.15.4/6LowPAN and Zigbee. Another advantage using the Firefly is that it supports multiple types of frequency bands such as 2.4 GHz, 915- and 920 MHz band. Radio parameters such as modulation, data rate and transmission power are highly configurable.

### 2.4.3 Raspberry Pi 3

The Raspberry Pi 3 is a single board computer developed by the Raspberry Pi Foundation [25]. It contains components suchs as WIFI, several USB ports, 1 GB RAM and a quad core ARM processor among several other features. Due to its relativly high performance for a low price, it has become a popular developing tool used in projects at home, in school and for academic research.

# 3 Experiment implementation/setup

The experiment testbed used for this project includes CCN-lite code implemented on Contiki-OS for the sensor nodes. Slip-radios is based on Raspberry Pis to act as border routers. Suitable communication software described in previous sections enables the transmitting of data between the border router and the sensor nodes. The topology of the network is to be considered a star network. This experiment layout, illustrated in figure [insert figure], can be viewed a normal communication scenario between a node and its border router.

- Show setup,
- A picture,
- how the program flows.
- imp The experimental implementation used in this paper includes CCN-lite application software running on Contiki-OS. The experimental implementation used in this paper can be divided into a two sections, the sensor and the border gateway. The sensor is runned by a modified CCN-lite application software that is running on Contiki-OS.
- Software used for Mote
- Mote hooked up on computer
- Sparrow
- Raspberry Pi, Software used on Computer As a border gateway, a Raspberry Pi3 A Raspberry Pi3 is used together with a slip-radio of which fully support As a border gateway, a Raspberry Pi3 is used together with a Zolertita[look up real name] slip radio. The slip radio fully supports communication over the 802.15.4 radio network. Together,
- communication flow.
- sensortags The IoT device used in this project is the CC2650 described in previous section. The node has a Contiki-OS version of the CCN-lite code

  The node has a CCN-lite implementation suitable for the Contiki-OS installed on it, providing the communication pattern suitable for this project.

## 3.1 Delimitations

Troughout this paper and the thesis project overall, power consumption has not been taken into consideration whatsoever.

# 4 ping/peek performance test 1

## 4.1 Purpose

The purpose of this experiment is to measure the roundtrip time, latency, between a sensornode and a border gateway using ping and CCN peek commands. The results show which of these alternatives has the lowest latency, how much they differ and if there is a common pattern between them. It is also interesting to see how much time it takes for a sensor node to consume an CCN interest. From a more overview perspectrive, it is very important that the processing time of a CCN interest does not take to long time or to much resources and that it should be feasible for a sensor to deal with. If the computation time of returning data is to large, then CCN would be considered not suitable to be used for a IoT device. From here on, latency and round trip time is used interchangeably aswell as CCN peek and peek.

[Make the nice picture and write about why that picture is interesting. This could be something in the beginning of the result section, the arguments could be around it. Showing the roundtrip time and processing time.]

- formulera purpose of this experiment, på en högre nivå, vad är det som motiverar varför dessa mätningar görs.
- Skriv om skillnaden i RTT pga av prints.

## 4.2 Exprimental setup

... In this experiment, a sensornode is connected via USB to a computer where one can monitor messages from the sensor. Through a 802.15.4 radio network, the sensor connect to a border router with Sparrow software running on it. All communication and message passing will be made between the the gateway and the sensornode over the 802.15.4 network. Above the 802.15.4 radio in the networking stack, data is encapsulated into 6LowPAN packets containing a full IPv6 header(of size 40 bytes). There are possibilites in Contiki-OS to compress the IP and networking headers by different strategies, but in these experiments only the uncompressed 40 bytes IPv6 header will be considered. Thereafter the application data is encapsulated by either UDP or ICMPv6 as ther transportation protocol, both of those headers consist of 8 bytes.

The roundtrip time will be measured using the ping6 command live tool which uses the ICMPv6 protocol. A similar tool has been developed to measure the latency for a CCN-peek request. Both tools are used in the similar way as in figure 4, where the requestor/consumer is on the left-hand side and the sensor/producer is on the right-hand side. Time is represented on the Y-axis going from the top of the figure to the bottom. The latency is measured in time units from the requestors perspective, it starts when the interest/ping has been sent from the requestor and stops when the requested data has been replyed from the sensor. [Although these latency tests are measured in time units from the requestors perspective, when possible, processing time will be conducted from the sensor when possible. This is to verify that the roundtrip values are valid. ]. As seen in figure 4, one can translate the roundtrip-calculation into the equation:

$$\textit{latency (roundtrip)} = \textit{interest transmission time} + \textit{data transmission time} + \textit{processing time} \quad (1)$$

$$\Leftrightarrow$$

$$latency\ (roundtrip) = 2\ x\ transmission\ time + processing\ time \tag{2}$$

$$\Leftrightarrow$$

$$transmission\ time = (roundtrip\ time - processing\ time) / 2 \tag{3}$$

where transmission time is the time it takes to transfer the data on the radio and processing time is the time it takes for the node to process the request. Queueing delay is possible in the system, such delay is here included under processing time.

In this test, the border router will ping, or peek, the sensor hundred times, thereafter the minimum, the median and average latency values are calculated from the result. The only variable that is changable in this experiment is the packet size of the outgoing data transmitted on the radio link towards the sensor.

Ping and peek differs how the total packet size transmitted over the radio is choosen. With the Ping approach, one adds an extra data payload in the request by setting a flag and assigning how much extra payload the packet size should contain. For Peek on the other hand, in order to change the packet size one has to adjust the naming of the data to a suitable length. The shortest name a data can have is just one single character (which equals to one byte). In this experiment, the length of the named data will be of $1 + 5 * X$.

There is an underlaying assumption that the size of the outgoing packets from the wireless radio has to be the same in both cases ping and peek. The reason is that the time spent on transmissioning the bits on the wireless link should be kept the same in both cases otherwise it could affect the result giving shorter roundtrip times for one system. Ping and peek use the same network protocols up until 6LowPAN(IP layer), the header sizes of UDP and ICMPv6 are the same at 8 byte each. But CCN has an application header of 16 bytes and the shortest name possible of the named data is one byte. To make a ping message equal to the CCN header and the name, the experiment will use a ping payload of $17 + 5 * X$.

Table 1 shows the mapping of the size of named CCN peek data, the ping payload and how big their total size will be on the 802.15.4 radio link. The total packet range is an interval of five bytes from 93 up to 148 bytes. Even though 802.15.4 has a MTU limit at 127 bytes for sending data into one frame on the wire, it can be interesting to see how the latency varies when fragmentated as well. Therefore latency measurements are of packet size up to 148 bytes.

## 4.3 Result

The results from the latency times with ping6 are shown in figure 5, where packet size, in bytes, sent on the radio link is shown on the x-axis and the latency time, in milliseconds, on the y-axis. The same axis layout holds for figure 6 and 7, but those figures shows the roundtrip time for CCN peek with and without debugging turned on. The results, illustradet in figure 5, show that the median latency for a ping6 request is very close to around 25 ms from 93 bytes in packet size up to 123 bytes. The median latency when sending a CCN peek request without debugging, shown in figure 6 is little above 25 ms for packet sizes the fragmentation limit. Roundtrip times when debugging is turned on, illustrated in figure 7, show a median of around

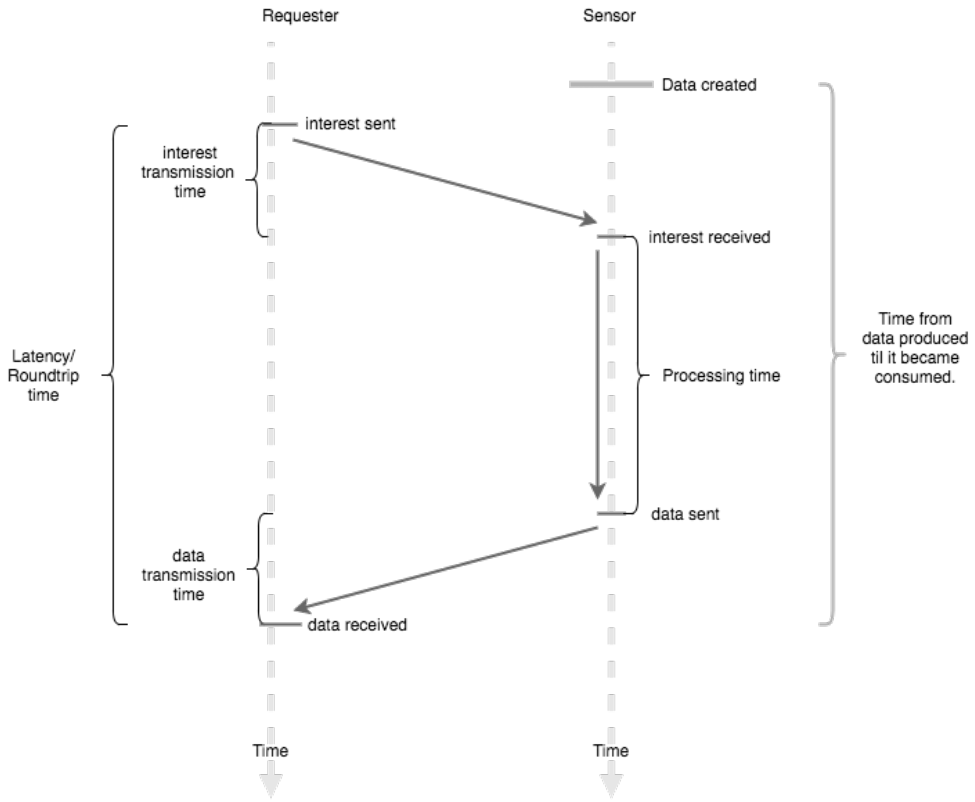


*Figure 4: The roundtrip measurement is the whole time it takes for a client to initiate a request, and get data as a response. Transmission time is the time the bits spent on the wire or radio. Processing time is calculated from the time an interest was received til it was sent back to the requestor.*

| CCN Peek | 1 (17) | 6 (22) | 11 (27) | 16 (32) | 21 (37) | 26 (42) | 31 (47) | 36 (52) | 41 (57) | 46 (62) | 51 (67) | 56 (72) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ping | 17 | 22 | 27 | 32 | 37 | 42 | 47 | 52 | 57 | 62 | 67 | 72 |
| 802.15.4 | 93 | 98 | 103 | 108 | 113 | 118 | 123 | 128 | 133 | 138 | 143 | 148 |

*Table 1: Row one shows the length of the named data in bytes, if the data is named "sensor" it is equal to six bytes. The number in the parenthethis show the size of the application, CCN header and length of named data included. Row two shows the data payload of a ping message, equivialent to the application data. Row three shows the total amount of data sent on the 802.15.4 radio toward the sensor node.*

50 ms. The 20-25 ms difference (around 100%) between the two peek results shows that there is a lot that can slow down the processing power, in this particular case it is only due to printing out massive amount of information toward the user in the terminal.

The CCN peek (from here on only concerning without debugging) latency time is only a few milliseconds apart from the ping counterpart when the same amount of data sent of the network. This indicates that, in a bigger perspective, the time it takes to process an incoming CCN interest is almost negligible for the sensornode although it is constrained. Although the difference is very small, when comparing latencies in figure 5 and figure 6 one can clearly see the small jump between them. This makes sense when considering that peek has to look up the content, process it and then respond to the requester, whereas ping would more or less respond direclty. Experiments to find out where most of the overhead time of CCN latencies went to, attempts of additional tests were done. Unfortunately the time resolution of the sensor node put an end to suchs experiements, whereas the sensor only have a resolution of 1/128 seconds (more of this later).

In all tests, the latency remain relativly flat even though the packet size is increasing, except the region around 123 byte to 128 byte. This indicates that the transmission delay has small to none overall effect on the latency. It also corresponds well to the fact that it theoretically takes 0.5 ms to send 127 bytes on a link that has a transmission rate of 250 kbit/s.

All measurements show a five to ten milliseconds difference between the minimium latency and the average/median latency. The histogram for the ping and peek latency measurements are illustrated in figure 8 and 9. They show the number of roundtrips, for application sizes of 17, 22 and 27 bytes, that can be categorised together and thereby see if there is any outliers that can be obmitted. Both histograms show that there is only a few such outliers and the absolute majority of the roundtrips lay around 24-28 ms. This inidicates that, even though the average and median latencies are close to each other, the median value is the correct way of measurement.

The fifteen to twenty millisecond jump in latency from 123 byte to 128 byte can be seen in all tests and is due to the 127 byte MTU of 802.15.4 which result in packet fragmentation at 127 bytes. The latency remains stable even after the fragmentation limit, which makes sense considering the flatness of the latency described above.

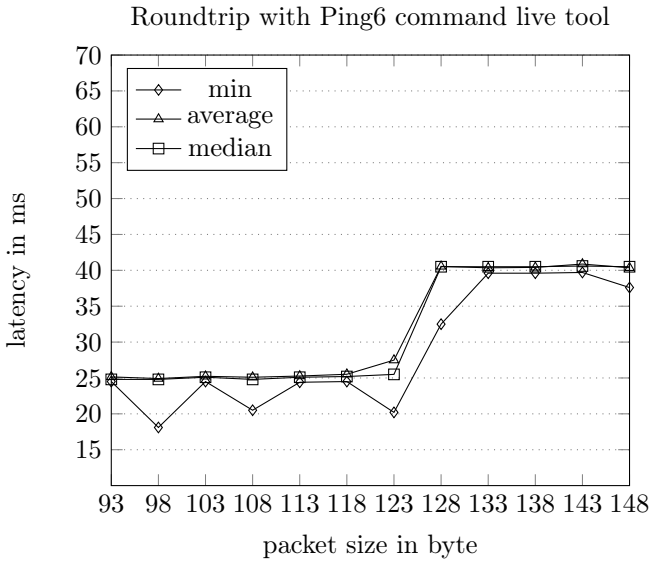


*Figure 5: Latencies in milliseconds when pinging a sensornode with packet sizes from 93 byte to 148 byte transmitted over the radio. The latencies stays stable at around 27 ms up until 123 byte. After fragmentataion, the latency raise and stay stable at around 43 ms.*

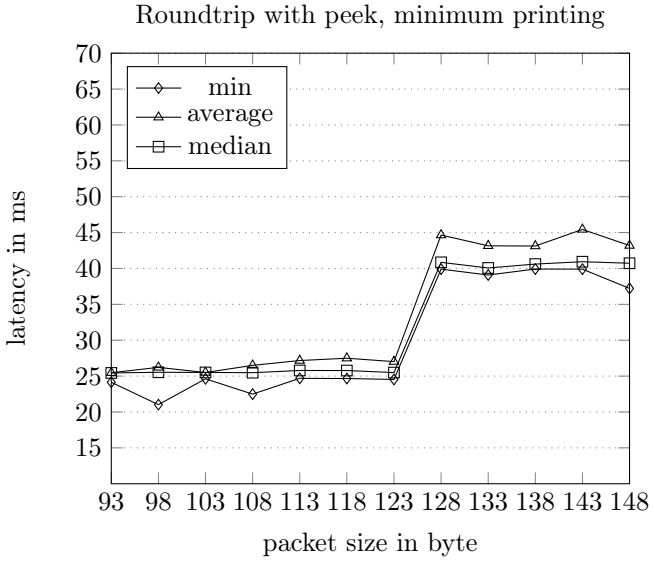


*Figure 6: Latencies in milliseconds for peek interest requess of sizes from 93 byte to 148 byte. The latencies stays stable at around 123 ms between 93 byte to 123 byte. After fragmentation, the latency raise and stays stable at around 135 ms instead.*

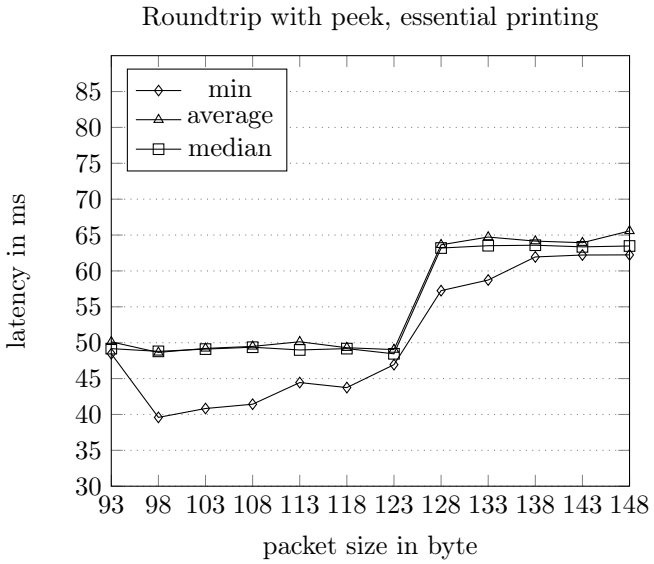


*Figure 7: Write same as with debugging, or is this graph necessary??*

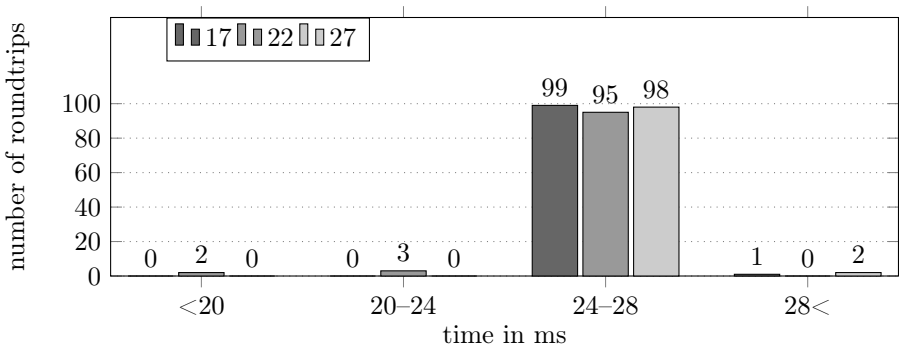


*Figure 8: Histogram of ping latencies result. A majority of the roundtrips end up in the 24-28 ms span.*

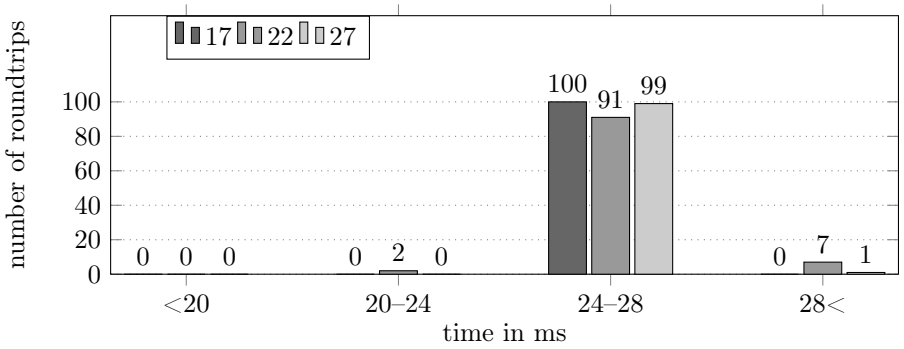


*Figure 9: Histogram of CCN-peek latencies result. Almost all of the roundtrips ended up in the 24-28 ms interval.*

# 5 Suitable evaluation test

## 5.1 Purpose

Follow sequence and periodicity from sensor

## 5.2 Method

Develop software

## 5.3 Result

Suitable.

# 6 Discussion

# 7 Conclusion

- Summarize your contributions
- Conclusions from the results
- Implications for the future
- Be breif!

## 7.1 Future work

# References


[1] Alan Carlton. Information-centric networking could fix these Internet problems kernel description, 2016.

[2] Bengt Ahlgren, Christian Dannewits, Claudio Imbreenda, Dirk Kutscher, and Börje Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7), 2012.

[3] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[4] 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. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049, 2014.

[5] Anders Lindgren, Fehmi Ben Abdesslem, Bengt Ahlgren, Olov Schelen, and Adeel Mohammad Malik. Design choices for the iot in information-centric networks. *IEEE Consumer Communications and Networking Conference*, 2016.

[6] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos. Information-centric networking for the internet of things: challenges and opportunities. *IEEE Network*, 30(2):92–100, March 2016.

[7] Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C. Schmidt, and Matthias Wählisch. Information centric networking in the iot: Experiments with ndn in the wild. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 77–86, New York, NY, USA, 2014. ACM.

[8] Riot. https://riot-os.org/ visited 2017-10-10.

[9] Yanqiu Wu. Adapting information-centric networking to small sensor nodes for hetrogeneous iot network. 2016.

[10] Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 1: Mac sublayer. *IEEE Std 802.15.4-2011 (Amendment to IEEE Std 802.15.4-2011)*, pages 1–225, April 2012.

[11] Ian Poole. Ieee 802.15.4 technology and standard. http://www.radio-electronics.com/info/wireless/ieee-802-15-4/wireless-standard-technology.php visited 2017-07-22.

[12] Internet Protocol, Darpa Internet program protocol specification. RFC 791, September 1981.

[13] Steve Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[14] Steve Deering and R. Hinden. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007.

[15] Mqtt. http://www.mqtt.org visited 2017-08-01.

[16] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*, pages 791–798. IEEE, 2008.

[17] Andy Stanford-Clark and Hong Linh Truong. Mqtt for sensor networks(mqtt-sn), version 1.2. http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf visited 2017-07-28, 2008.

[18] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7959, June 2014.

[19] Ccn-lite, content centric netowrking lite platform. http://www.ccn-lite.net/ visited 2017-08-20.

[20] Contiki-os. http://www.contiki-os.org visited 2017-07-24.

[21] 6lbr. http://cetic.github.io/6lbr/ visited 2017-08-20.

[22] Sparrow application layer. https://github.com/sics-iot/sparrow visited 2017-08-20.

[23] Texas instruments, cc2650 sensortag. http://www.ti.com/tool/cc2650stk visited 2017-09-20.

[24] Zolertia firefly. https://zolertia.io/product/firefly/ visited 2017-09-20.

[25] Raspberry pi 3. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ visited 2017-09-20.