Design, Development and Deployment of an Information-Centric Networking based solution for the Internet-of-Things

Course Leader:
Edith Ngai

Supervisors:
Adeel M. Malik (Ericsson)
Börje Ohlman (Ericsson)
Bengt Ahlgren (SICS)
Anders Lindgren (SICS)

August 26, 2016
1 Introduction

With a rapid increase in media generation and consumption in the past few years, user demand patterns on the internet have changed substantially. Today internet traffic is predominantly video traffic. By 2021 almost 70% of all mobile data traffic will be from video [1]. On the other hand emerging technology trends such as the Internet-of-Things (IoT) are expected to have a huge impact in the Information and Communication Technology (ICT) arena giving birth to new technology and business opportunities such as smart cities, smart metering, industry automation etc. According to the Ericsson Mobility report released in June 2016 [2], the total number of connected devices is expected to be around 16 billion by 2021.

The fast evolving user demand and traffic patterns pose new challenges which inspire the need for more efficient networks that can deliver content faster with minimal overhead. Information-Centric Networking (ICN) is a new networking paradigm where the communication is based on named content unlike traditional host-centric IP networking where it is based on named hosts. The ICN concept was first presented in the Content Centric Networking proposal of Jacobson et al. [3]. ICN is deemed to be more in line with the way Internet is used today where users, in most cases, are interested in content and not its location. ICN treats content as a primitive and therefore has the potential to facilitate the current internet usage patterns more effectively.

IoT is one such application area where the user is mostly interested in data. While ICN is a promising evolutionary direction for the internet, it has its own set of challenges in the IoT context such as naming data, handling data streams, handling mutable data entities, name-based routing, searching data, handling actuators etc [4].

Over the years ICN has been realized in the form of several different protocols in various research efforts. Some of them are Content-Centric Networking (CCN), Named-Data Networking (NDN), NetInf (Network of Information), PURSUIT etc [5]. This project is based on CCN [6], a protocol that employs name-based routing and hierarchical names to address content.

The overall goals with Project CS 2016 are the following:

- Take advantage of ICN technology in the IoT sensor domain by directly mapping sensor readings to named data in the network, and showing the feasibility of some of the ideas in [4].
- Using the ICN-based IoT infrastructure, develop one or a few example Android apps making use of the sensed data via ICN transport.
- Get experience from working in a team in a software development project towards common goals.
2 Background

The following sub-sections give a background to ICN, the CCN and MQTT protocols and the GreenIoT project.

2.1 ICN and CCN

Information-centric networking (ICN) [5] is a communication paradigm based on *named data*. Entities communicate by *providing* or *requesting* named data without regard to its location. The data is thus made independent of location, which results in that any node in the network can provide any piece of named data to other nodes, making it possible to cache data in the network without violating the assumptions of the network architecture. The information-centric networking paradigm was developed with the goal to eventually replace the current Internet, and is therefore sometimes called a ‘Future Internet’ architecture.

In contrast, current network technology, including the Internet, is *host-centric*. Hosts are named with some kind of addresses, for example, IP addresses, and communication is carried out by sending messages, or setting up connections, between named hosts. The actual data being transferred is anonymous at the network level.

Content-centric networking (CCN) [3] is a particular implementation of ICN that was initially designed by Van Jacobson at PARC. Today with CCN 1.0 [6] the details of the design has evolved to some extent, for example, making it more suitable for implementation in a high-speed router.

The CCN protocol is quite simple with two main message types: *interests* and *data*. In order to retrieve data, a client sends an interest message, specifying the name of a named data object. CCN uses a hierarchical naming scheme not very different from the URLs we see on the web. The routers in the CCN network forwards the interest message towards publishers of the specified name. Any router on the path can respond with the matching named data object. The message with the data is routed back to the client using temporary state in a ‘pending interest table’ that was set up in each router on the path when forwarding the interest.

2.2 MQTT

MQTT is a light-weight messaging protocol with a publish/subscribe interaction model that runs on top of TCP/IP. A *broker* keeps track of subscriptions made by interested clients, and is responsible for redistributing incoming messages according to the subscriptions. The publish/subscribe model is topic-based, which means that clients subscribe by specifying a topic filter which then is used to match the topic of incoming messages with. MQTT is simple enough to be
implemented in constrained sensor devices. An implementation is, for example, available for Contiki OS [7]. In the IoT domain, the protocol can be used for sending sensor readings from a sensor device to a broker that redistributes the data to a set of clients. Some good reading resources for MQTT are the MQTT specification [8] and the HiveMQ online tutorial [9].

2.3 GreenIoT project

Project CS 2016 is carried out in the context of ‘GreenIoT’\footnote{https://www.sics.se/projects/greeniot-an-energy-efficient-iot-platform-for-open-data-and-sustainable-development}, a project funded by Vinnova, the Swedish Innovation Agency. Project partners are Uppsala University, Royal Institute of Technology (KTH), SICS, Uppsala Kommun, Ericsson, IBM, SenseAir, Upwis, and 4Dialog.

The overall goal of GreenIoT is to create an environmental sensing system and application platform for the public sector and private parties to develop practical and innovative applications based on open data for sustainable city development. The open data generated from the platform also enables citizens to make green decisions in their daily life. A major goal of the project is an integrated solution for an environmental sensing system, which enables experimentation with applications and services using open environmental data, in particular for sustainable urban and transportation planning. The GreenIoT architecture is manifested in terms of a testbed in Uppsala.

The sensing system and application platform are built from unique technology that provides open interfaces at several levels, energy and resource efficiency, and application independence. The project uses a unique tool for visualisation in four dimensions, which seamlessly integrates with sensor data for real-time feedback. The GreenIoT testbed and the open data allows third parties to develop and test new sensing products and solutions that could be exported to international markets, such as China and India.

The project is currently deploying sensors for measurement of air pollution in the Uppsala city centre. The sensors run the Contiki operating system and provide data using the MQTT protocol. The intention is to use this data from the GreenIoT sensor deployment in Project CS 2016 in addition to data generated by an experimental deployment of CCN sensors.
3 System Description

Fig. 1 shows the system setup which is split into two parts. One that will be developed by students in this project and the other that will be developed in the GreenIoT project.

Sensor devices produce data which is either pushed by the sensors towards the network or pulled by network nodes from the sensors. The overall system has both CCN and MQTT sensors. In this project, users will connect to the sensors through a CCN network which includes CCN routers and a CCN-MQTT gateway. The CCN-MQTT gateway facilitates data injection from the MQTT domain to the CCN domain and vice versa, thereby making data from MQTT sensors available to users in the CCN domain and data from CCN sensors available to users in the MQTT domain.

In the GreenIoT project, data produced by the MQTT sensors will be stored in the IoT datastore. The IoT datastore serves as long-term storage for all raw and processed sensor data. One goal of this project is to use the same datastore for the CCN domain. This can be achieved using a CCN interface to this datastore.

The CCN network also hosts a sensor directory service which keeps information of all sensor devices registered in the system. It also stores information
functions available in the CCN network such as a mean of all readings produced in a certain time interval. The sensor directory service will also facilitate auto-configuration of sensors and network nodes in the system.

3.1 System components

The following sub sections discuss the details of each system component.

3.1.1 Sensors

The overall system will have two types of sensors, CCN and MQTT. As mentioned in section 2.3, the MQTT sensors are part of the GreenIoT project and can serve as data sources for this project. In this project the students will primarily work on an experimental deployment of CCN sensors and tackle the associated challenges.

CCN sensors will connect to the CCN network through a sensor gateway. The default interaction model that CCN offers is pull-based i.e. CCN Interests are sent to the CCN sensors that respond with the requested data. In contrast to the CCN interaction model, MQTT sensors push their data to an MQTT broker using an MQTT Publish message.

3.1.2 CCN-MQTT gateway & MQTT broker

The CCN-MQTT gateway connects the CCN domain with the MQTT domain. It allows flow of sensor data between the two domains. This can be achieved by subscribing from the CCN-MQTT gateway to the MQTT broker (using an MQTT Subscribe message) and by subscribing from the MQTT broker to the CCN-MQTT gateway. The MQTT broker acts as a single point of contact between the MQTT sensors and the outside world.

The CCN-MQTT gateway subscribes to the MQTT broker for various MQTT topics that correspond to data produced by the MQTT sensors. To subscribe to all the topics registered at the MQTT broker, the CCN-MQTT gateway must subscribe to the topic name “#”. Readings generated by the MQTT sensors are pushed by the MQTT broker towards the CCN-MQTT gateway using MQTT Publish messages. Subsequently, the CCN-MQTT gateway will publish these messages in the CCN domain.

The MQTT broker is not directly a part of this project and the students are not expected to work with it. However it is relevant for the project as a point of interaction with the MQTT sensors. The GreenIoT project will have designated MQTT brokers which the CCN-MQTT gateway must communicate with. For data to flow from the CCN sensors to the MQTT domain, the MQTT broker should also
subscribe to various MQTT topics at the CCN-MQTT gateway that correspond to streams of data being produced by the CCN sensors. Mapping CCN names to MQTT topics and vice versa is a challenge that should be solved in this project.

### 3.1.3 CCN routers

The CCN network contains CCN routers which provide a forwarding plane to facilitate connectivity between the different system components. These routers will also cache data produced by the sensors when responses to CCN Interests traverse through them. Since IoT data is really small in size, a “cache all” policy will suffice. A simple cache eviction policy should also be implemented to evict the oldest readings or the least recently used readings.

The forwarding plane of the CCN routers should be auto-configured. Forwarding entries for sensors that attach to the system should be installed automatically in the CCN routers in the network. This can be facilitated by the sensor directory service which keeps track of sensors registered in the system.

### 3.1.4 CCN datastore interface and IoT datastore

Data generated by sensors can be stored in several places in the network including the sensors, CCN routers and the user device. However, a long-term reliable storage is needed which can guarantee access to all historical data. The IoT datastore can potentially also store processed data e.g. maximum CO₂ emission level in the past one year.

The IoT datastore is part of the GreenIoT project and provides an HTTP interface. To use the IoT datastore for the CCN domain a CCN interface to the datastore is needed. A CCN datastore interface should be developed as part of the project. This interface will allow data to be fetched from the IoT datastore for the users in the CCN domain and also for data to be injected from the CCN domain to the IoT datastore. This will also enable the GreenIoT users to use data produced by the CCN sensors.

### 3.1.5 Sensor Directory Service (DS)

A Sensor Directory Service (DS) will keep track of all the sensors registered in the system. For each registered sensor it will also store the static metadata of the sensor e.g., CCN prefix for the sensor, sensor type (CO₂, temperature etc.), sensor id, sensor device model, geo-location, reading unit (ppm, °C) etc. The upside of saving sensor metadata in the DS is that the sensor will only need to send this data once. Users can retrieve this data from the DS.
The DS should also keep track of Named Functions [10]. This is discussed in more detail in section 3.4. Named Functions can be registered in the DS so that users can access the DS to see the list of operations available.

The DS should also facilitate auto-configuration. The DS can assign CCN prefixes to the sensor devices dynamically. On registering with the DS, a sensor device can be assigned a CCN prefix under which it will make its data available. Following prefix assignment the DS should install forwarding entries in the CCN routers for the assigned prefix.

3.1.6 CCN Android application

One of the main goals of this project is to develop a CCN Android application which users can use to access data produced by the sensors. The application should be able to query the Sensor Directory Service (DS) to see what sensors and named functions are registered in the system.

3.2 Naming sensor data

Both CCN and MQTT have hierarchical names. However the difference in their naming arises from the fact that MQTT is a message queuing protocol while CCN is not. In MQTT, every reading produced by a sensor corresponds to the same MQTT topic or name whereas in CCN each sensor reading will have its own unique name. This raises challenges in mapping between MQTT topics and CCN object names which is needed when transferring sensor data between the CCN and MQTT domains.

How CCN sensor data is named has implications that influence how the entire system functions. It also affects how the user application functions. Some challenges with regard to data naming are listed as follows:-

- Should a CCN name map only to immutable objects or can there be CCN names that map to mutable objects too.
- Should the CCN object namespace map to a sensor or a sensor stream?
- How are forwarding entries for object names installed in the CCN routers? Should an entry be installed per CCN object, per sensor data stream or per sensor?
- How will a user know the names of objects published e.g., at a certain point in time? How will a user request the latest reading of a sensor?
• What names are stored in the information of a sensor in the sensor directory service? Should the sensor directory service only keep track of the sensors registered or should it also keep track of all the data objects produced?

• Should an identical object stored in the IoT datastore have the same name? If yes, how does the forwarding plane in the routers ensure that an Interest is forwarded towards the IoT datastore if not satisfied elsewhere?

• Name mapping between MQTT data and CCN objects.

The above mentioned challenges highlight the fact that data naming is an architectural issue. The naming approach employed should neither be too specific to an application, in which case it will be rendered unusable for other applications, nor be too general in which case the applications will be left to do a lot. Making concrete assumptions about the naming approach has been deliberately avoided in this specification as this is a problem best tackled while working through the different challenges of the project.

3.3 Sensor data representation and metadata

Sensor data and metadata should be represented using the Sensor Markup Language (SenML) which defines media types for representing simple sensor measurements and device parameters [11]. Metadata can be static (remains constant for a sensor) e.g. sensor id, sensor device model etc. or dynamic (different for each sensor reading) e.g. timestamp, geo-location for moving sensors etc. As mentioned in section 3.1.5, static data can be stored in the sensor directory service to avoid recurring transmission from resource constrained sensors.

3.4 IoT data storage and processing

Data can be stored and processed at several different places in the system. It can be stored at the following locations in the system:

1. CCN sensor: Sensors can offer very limited storage due to memory limitations. They should only store data if really needed. An example of this is a sensor which only generates readings periodically and not when an Interest is received. On receiving an Interest for the current reading the sensor returns the last produced reading. A sensor like this can store historical readings locally.

2. CCN network: CCN routers in the network can have high capacity disks to cache sensor data. Since sensor data will be in the order of a few bytes
or tens of bytes, such storage media should be able to store a huge amount of sensor readings. However, CCN routers do not offer guaranteed storage of sensor data. Also they only act as on-path caches and therefore a CCN router that does not lie in the path of a CCN Interest cannot serve the Interest even if it contains the requested data.

3. IoT datastore: Unlike CCN routers, the IoT datastore can offer guaranteed storage of raw and processed sensor data. The IoT datastore should store data from both CCN and MQTT sensors.

4. User device: Caching is an inherent feature of ICN. Therefore the CCN client running in the user device should also cache sensor data. Caching data in the user device enhances application performance and eliminates the need to request the cached object from the CCN network.

Sensor data can be processed in a number of ways and places in the network. Some possibilities are listed as follows:-

1. CCN sensor: In some cases carrying out simple processing tasks at the sensor can save its transmission cycles e.g. the mean of ten readings can be calculated by the sensor in order to avoid having to transmit all the ten readings towards the network. This assumes that the sensor should have stored some historical data locally.

2. Named-Function Networking (NFN) [10]: This is an area that should be explored during the course of this project. NFN offers a way to address pre-defined functions hosted in the network. The idea of NFN is to offload a defined data processing operation from the requester to another node in the network so that the requester does not need to worry about how the operation is performed. An example of this is to make forecast of the CO$_2$ level in the air at a future point in time. Hosting Named Functions in the network opens up a great deal of possibilities for application developers that can have homogeneous access to a rich set of ready-to-use functions.

3. User device: The user device can also process raw sensor data to produce processed data relevant for the application. However, these processing tasks should not be too heavy in the interest of application performance.

### 3.5 Auto-configuration

One goal of this project is that the sensors should be able attach to the network without any manual configuration. This can be facilitated by using the Sensor
Directory Service (DS). When a sensor is turned on, it should register itself with the DS. In order to register with the DS the sensor should provide it with all metadata e.g. type, id, device model, geo-location, reading unit (ppm, °C) etc. The DS should provide the sensor with a CCN prefix dynamically on which the sensor will listen to serve incoming Interests. The DS should also install forwarding entries in the CCN routers for the assigned prefix, hence auto-configuring the forwarding plane of CCN.

### 3.6 Sensor interaction model

Figure 2 compares the interaction models of CCN and MQTT. CCN has a pull-based interaction model where a client sends an Interest message in order to retrieve data from the CCN sensor. In contrast to CCN, MQTT has a push-based interaction model. Clients subscribe to the MQTT broker once. Consequently, any new messages arriving at the MQTT broker from the sensors that match the subscription are pushed out by the MQTT broker towards the client.

In other words, CCN does not have a Publish-Subscribe mechanism. However, CCN is a protocol still under development. If there is good motivation to introduce Publish-Subscribe for CCN to achieve the goals of this project, the students are encouraged to do it.
4 Project Goals

This section summarizes the goals of this project. The project goals are mentioned as follows:-

- Develop sensor software and deploy these sensors to sense different types of data.

- Develop an Android CCN relay service and an Android application that makes use of sensor data. The Android application should be able to see registered sensors and named functions in the network.

- Develop CCN router software and deploy them. Implement a simple cache eviction policy.

- Develop and deploy a CCN-MQTT gateway to interact with the CCN network on one side and an MQTT broker on the other. This gateway should facilitate data flow in both directions.

- Develop and deploy a CCN datastore interface which interacts with the CCN network on one side and the IoT datastore on the other side using an HTTP interface. This interface should allow for data insertion from the CCN domain to the IoT datastore and data retrieval from the IoT datastore to the CCN domain.

- Develop and deploy a sensor directory service which stores information of registered sensors and named functions and also facilitates auto-configuration of sensors and the forwarding plane in the CCN network.

- Develop and implement a naming scheme for CCN objects. Also define how name mapping will be done between CCN objects and MQTT data.

- Implement Named-Function Networking (NFN) and host named functions in the network which are used by the Android application to execute a specific task.
5 Software and Tools

Figure 3 gives an overview of the software stack in the different network components. From the CCN client to sensor gateway, CCN will run over TCP or UDP. The interface between the sensor gateway and the CCN sensor is over 802.15.4 radio. Here CCN can either run directly on 802.15.4 link layer or run over UDP/IP over 6LoWPAN over 802.15.4. The advantage of the former is that the network stack will be much lighter and will give leeway to perform more sophisticated functions on the sensor platform.

Students are not expected to work with the MQTT sensors. However familiarity of the stack would be helpful in developing the CCN-MQTT gateway.

The following text lists the relevant software and software requirements for this project:

- Sensor platform: For the sensor the two IoT operating systems relevant are the RIOT OS [12] and the Contiki OS [7]. The open source implementation of CCN, CCN-lite [13], has been ported for both the RIOT OS and the
Contiki OS. These ports will be available for the students to start from. Based on the requirements of the project these implementations must be enhanced to include the needed features.

- **CCN routers**: There are two open source implementations of CCN that can be used for the CCN routers. CCN-lite [13], which is a lightweight implementation of CCN and CCNx Distillery [14] which is a more elaborate implementation of CCN from PARC. It is recommended to use CCNx Distillery for the CCNx routers since the implementation provides a richer API and has good support on the developer mailing list.

- **CCN Android application**: For the Android application a CCN client must be implemented. A basic version of CCN client is available to use for Android. A CCN relay must also be implemented as an Android “service” that runs independent of the application. This will allow other CCN applications to also make use of this service. For the CCN relay it is recommended to use native CCNx Distillery or CCN-lite code and build it using Android Native Development Kit (NDK). Students must also design and implement a front-end for the application.

- **CCN-MQTT gateway**: Two basic implementations of the CCN-MQTT gateway are available to use for this project. One written in C and the other in Python. MQTT client implementations are available from the Eclipse Paho project [15].

- **Sensor Directory Service (DS) and CCN datastore interface**: These should be developed by the students from scratch.
6 Deliverables

The deliverables of this project are mentioned as follows:-

- **Software repositories:** Android application, Android CCN relay service, CCN router software, CCN sensor software, CCN-MQTT gateway software, sensor directory service software, CCN datastore interface software. Students must use Git for all software repositories.

- **Project report:** This report should discuss the system architecture and components. It should also discuss system solutions that address different goals of the project on a conceptual level without going into implementation details e.g., object naming, data retrieval mechanisms, data storage and processing mechanisms, auto-configuration, data exchange between the CCN and MQTT domain, data formats and metadata, Named-Function Networking (NFN) etc.

- **Software report:** A report that describes implementation details of the different software components developed during the course of the project. This should include details such as software architecture, APIs used, software limitations, installation and usage guides, what was actually developed and what was resused etc.
References


