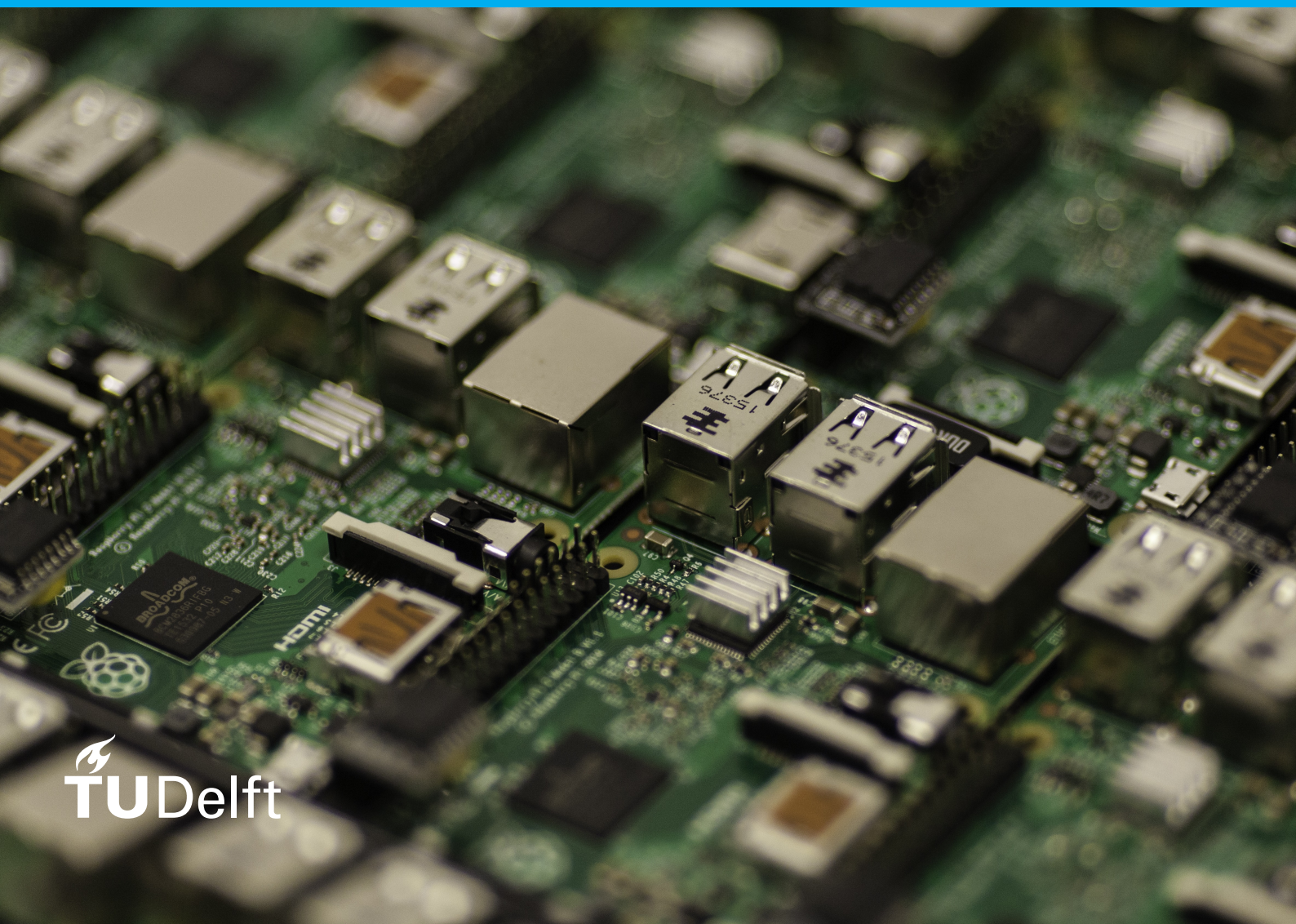
Assessment of Benefits and Drawbacks of ICN for IoT Applications

F.B. Drijver

Master thesis
Electrical Engineering
Faculty EEMCS



TU Delft

Assessment of Benefits and Drawbacks of ICN for IoT Applications

by

F.B. Drijver

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday, March 23, 2018

Student number	4217349	
Faculty:	Electrical Engineering, Mathematics and Computer Science (EEMCS)	
Master programme:	Electrical Engineering	
Specialisation:	Telecommunications & Sensing Systems	
Project duration:	Dec 1, 2016 – April 11, 2017	
	May 15, 2017 – March 23, 2018	
Thesis committee:	Dr. R. Litjens M.Sc.,	Associate Professor, TU Delft (supervisor)
	Dr. Ir. F.A. Kuipers,	Associate Professor, TU Delft
	Dr. Ir. L. D'Acunto,	Research Scientist, TNO (daily supervisor)
	K. Trichias M.Sc.,	Senior Engineer, Incelligent (daily supervisor)

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This thesis completes my master Electrical Engineering at Delft University of Technology. The goal of this thesis is to get a clear understanding of the main benefits and drawbacks of using Information Centric Networking (ICN) in Internet of Things (IoT) applications, compared to traditional IP deployments. Multiple scenarios are defined based on realistic IoT use cases, which are used to evaluate the performance of ICN as networking paradigm for IoT. We take an experimental approach in the performance evaluation, by conducting extensive network simulations using the NS-3 simulator.

The work for this thesis was carried out as a graduation project at TNO in The Hague, The Netherlands. TNO is a non-profit knowledge organisation, focusing on research in applied sciences. I spent twelve months at TNO's Networks department and worked there with many talented, smart and friendly people. I ended up at TNO with the help of my TU Delft supervisor Remco Litjens. After completing a summer internship, I asked Remco if he had any graduation projects available. A few days passed, after which I received several enthusiastic e-mails with thesis proposals from multiple members of TNO's Networks department. I found the proposal covering ICN for Internet of Things applications the most interesting. After an interview with my TNO supervisor Kostas Trichias and a first meeting with the Networks research manager, I was able to start working on my thesis on the December 1, 2016.

During my time at TNO, I liked the enthusiasm and passion of all members of the department. This graduation internship allowed me to reflect on my personal strengths and weaknesses, which helped me with my personal development.

Unfortunately, my TNO supervisor Kostas Trichias decided to leave TNO as of August 1, 2017 to return back to his home country for a new adventure. Thankfully, Lucia D'Acunto was willing to take over his role as my supervisor. Therefore I would really like to thank Remco Litjens, Kostas Trichias and Lucia D'Acunto for supporting me with this thesis.

F.B. Drijver
The Hague, February 8, 2018

Contents

List of Acronyms

ABE	Attribute-Based Encryption
AMQP	Advanced Message Queuing Protocol
CoAP	Constrained Application Protocol
ARP	Address Resolution Protocol
AS	Autonomous System
AWGN	Additive White Gaussian Noise
BRITE	Boston university Representative Internet Topology gEnerator
CCN	Content Centric Networking
CDN	Content Distribution Network
CS	Content Store
DNS	Domain Name System
FFD	Full Function device
FIB	Forwarding Information Base
FMC	Fixed and Mobile Converged networks
GRMR	Greedy Regional Multicast Routing
HTTP	Hypertext Transfer Protocol
ICN	Information Centric Networking
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
KDE	Kernel Density Estimation
KPI	Key Performance Indicator
LR-WPAN	Low-Rate Wireless Personal Area Networks
LRU	Least Recently Used
MAC	Medium Access Control
MDMR	Max Diversity Most Recent
MQTT	Message Queuing Telemetry Transport
MTU	Maximum Transmission Unit
NDN	Named Data Networking
NDP	Neighbour Discovery protocol
NFD	Named Data Networking Forwarding Daemon
NS-3	Network Simulator 3
NS3-DCE	NS3 with Direct Code Execution
NSF	American National Science Foundation
OQPSK	Offset Quadrature Phase-Shift Keying
OSI	Open Systems Interconnection (model)
PAN	Personal Area Network
PIT	Pending Interest Table

RAM	Random-Access Memory
RDC	Radio Duty Cycling
REST	REpresentational State Transfer
RFD	Reduced Function Device
RIP	Routing Information Protocol
RIPng	RIP next generation
RONR	Reactive Optimistic Name-based Routing
RV	rendezvous
SAAS	Sensing As A Service
SCS	Shared Caching System
SDN	Software-Defined Networking
TCP	Transmission Control Protocol
TLV	Type-Length-Value
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VIF	Vanilla Interest Flooding
XMPP	Extensible Messaging and Presence Protocol

1 Introduction

This chapter will present the main research goals of this thesis, by firstly introducing the concepts of Information Centric Networking (ICN) and the Internet of Things (IoT). A short description on the background of ICN will be given in Section 1.1. The basic concepts of ICN will be presented in Section 1.2. Section 1.3 will introduce the IoT by outlining the key characteristics of IoT deployments and applications. A qualitative comparison of ICN and IP is performed in Section 1.4.2, which is used to provide a first hypothesis into the main drawbacks and advantages of using ICN compared to classic IP technologies. This chapter concludes with a description of the main research questions in Section 1.5 and a thesis outline in Section 1.6.

1.1. Background

The American computer scientist Van Jacobson is renowned for his contributions to internet research which has had a big impact on the internet we use today. The most prominent example is his work on the TCP flow control mechanism [1], which is widely used in the current internet. In 2006, Van Jacobson presented his view on the future of the current IP oriented internet architecture by introducing the fundamentals of the Information Centric Networking (ICN) paradigm [2, 3]. According to Van Jacobson, the current host centric internet architecture is no longer fitting the data-centric way we use the internet today. Internet users are no longer interested to solely connect to a certain network resource, as was the case when the IP protocol was developed. Users nowadays typically want to receive certain data as fast as possible, caring less where the data is exactly stored. A data centric internet approach would allow hosts to request content based on a known data name, rather than on a host name, eliminating the need to discover the exact location where the data is stored.

The deployments of Peer-to-Peer (P2P) and Content Distribution Networks (CDN) indicate the need for a more data centric usage of the current internet by decoupling the data retrieval process from a specific location or server. In P2P networks, data is distributed over several 'peers' or nodes and a user can retrieve (parts of) the content from other peers. The original data source only needs to inject the content once into the P2P network, where the data is then distributed over several other nodes. CDNs are widely used by data services such as YouTube and Facebook. The main goal of CDNs is typically to move content closer to the end-user and therefore optimising network and service performance. This can be done by, for example, geographically distributing copies of the content over several severs or by deploying web caching, and thereby reducing the content retrieval delay and total network bandwidth usage.

The wide scale adoption of CDNs and P2P networks shows that we use the internet in a different way than originally was envisioned when the main foundations of the internet were developed. Both technologies can be seen as manners to map the current internet usage patterns to the legacy host centric IP based internet. ICN revises this approach by replacing the current internet implementation by a new ICN stack with native support for data centric networking, and thereby simplifying the required internet stack.

1.2. Information Centric Networking (ICN)

We will introduce the concept of ICN by addressing the most important building blocks of this new networking paradigm in Section 1.2.1. Moreover, a short description will be provided on the most popular ICN software implementations in Section 1.2.2.

1.2.1. ICN basic principals

We distinguish five main features in the ICN architecture. We start by explaining the used ICN terminology of defining consumers and producers. Afterwards, we will address how data can be named and how this naming scheme is used to forward data. The third feature we will cover is seen as one of the most promising features of ICN, namely the ability to cache named data on a network-wide scale. The last identified feature is the support for packet level security.

The roles of the nodes in the architecture

In ICN, nodes may take one out of two different types of roles: a consumer role or a producer role. A node which wants to receive a certain piece of data is called a consumer and a node which is able to supply new data into the network is called a producer. The assignment of consumer and producer roles is only fixed during one single data exchange. A node which, for example, currently has a producer role, is thus free to take a consumer role in the next data transmission.

Message types

In ICN only two packet types are envisioned; the Interest message and the Data message, which are shown in Figure 1.1. The ICN architecture uses a pull-based, 'one-Interest, one-Data' approach: consumers request for data by issuing an Interest message, and will receive at most one Data message matching that Interest. This Interest message contains the name of the data, which is used to forward the message towards the data producer or towards a cache, and a unique 'nonce' to detect duplicate Interests. Further, the selector field can be used to enforce additional constraints on the requested Data message. As an example, the *exclude* selector field can be used to exclude Data messages with certain name components which can be defined by the consumer.

Another node can *satisfy* this Interest by returning a packet containing the Data message, if it holds a copy of the specified content. If an intermediate node is not able to supply the specified data, then the Interest will be forwarded in the direction of the producer or in the direction of another cache. The Data message must at least contain the data name and the data itself, and may optionally be signed by the data producer. A consumer can use the signature and signature info to verify whether the data originates from a trusted producer. It must be noted that the specifics of the Interest and Data message formats are implementation dependent. Some ICN implementations may offer extra fields in the ICN messages.

Naming and forwarding

The data naming scheme typically follows a hierarchical structure, where different levels of this hierarchy are separated by a forward slash. An example ICN name for a picture can be: */TNO/Floris/Cat.jpg*. This hierarchical structure is used in the forwarding logic of the ICN hosts, where longest prefix matching is used to find a suitable next hop for Data and Interest messages. The ICN architecture defines a

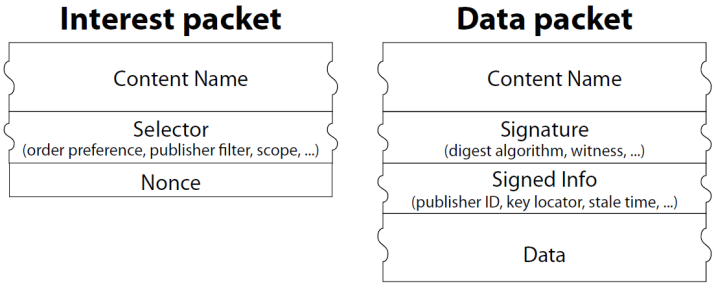


Figure 1.1: The ICN Interest and Data messages (packets), specified by [3]. Picture from [3].

modular forwarding mechanism on every ICN host which comprises a Content Store (CS), a Pending Interest Table (PIT) and a Forwarding Information Base (FIB). Data and Interest messages enter the ICN forwarding logic via so-called faces. A face can for example be an Ethernet interface or even a TCP or UDP socket. When an ICN host receives an Interest on a certain face, it will initially check whether it holds a copy of the data of interest in its CS. If so, it will return it to the requesting face via a Data message. Otherwise, the ICN host will check whether it had previously received any other Interest for the specified content. This check is done via a lookup in the PIT where each Interest is registered with a single entry containing the name and incoming face. If there is no matching entry in the PIT, a new PIT entry will be added for the Interest and the FIB will be consulted to find an appropriate face to forward the Interest to. If the PIT already contains an entry for the data requested by the incoming Interest, then the incoming face of this Interest will be added to that specific PIT entry, if it is not there yet.

When a Data message arrives at a node, the PIT will again be consulted to find the incoming faces of the corresponding Interest messages. The Data message will then be forwarded to all faces listed in the matching PIT entry. An overview of the ICN forwarding logic can be seen in Figure 1.2.

In-network caching

As mentioned earlier, ICN Data messages can be identified by their name. Naming Data messages based on the data that they carry rather than the host they are directed to enables caching at intermediary nodes on the data path. ICN hosts and routers can simply cache Data messages to satisfy future Interests for the same data. Caching named data could result in a reduced usage of network resources and faster data retrieval, since Interest messages could be satisfied with Data messages cached close to the interested node.

Packet-level security

Due to the simple ICN message structure with only Interest and Data messages, it is possible to use so-called content-based security. In the current IP-based internet, safe data transfer is guaranteed by using secured connections. A secure connection then requires two steps: establishing trust in the endpoint and securing the data transfer with cryptographic key pairs.

In ICN a different approach is followed. Instead of securing a connection, the Data messages themselves are being secured. It is therefore no longer necessary to set-up a secure connection between the requester and the producer, since the Data messages themselves can be verified. The signature in the Data message can be used to check that the content name, data and signed info are originating from the trusted producer. One of the main advantages of content-based security is the ability of reusing Data messages when secured data transfer is required. Content based security allows to cache signed Data messages, which would not be possible with the end-to-end secured connection in IP. Next to signing Data messages, it is also possible to encrypt the data itself, if the data should not be visible for other users. The fact that data names in Interest and Data messages are transmitted in the clear may pose some confidentiality issues when using ICN. A possible workaround to this may be that of using a hash of the human-readable name components as the Data name [5].

1.2.2. ICN implementations

Van Jacobson's proposal resulted in the development of multiple ICN software implementations. The two most popular implementations following the ICN design are called NDNx [6] and CCNx [7]. CCNx is the oldest of the two, dating back to 2006. Van Jacobson originally started the development of the CCNx software at the Palo Alto Research Center (PARC), which later acquired a trademark on the name CCNx. The multi campus NDN research project funded by the American National Science Foundation (NSF) started in 2010 and used the CCNx implementation as basis for their research. Starting from 2013, the implementation from the NDN project and the implementation from PARC have begun to diverge. The implementation developed by the NDN project was then renamed to NDNx. In february 2017 it was announced that Cisco acquired PARC's Content Centric Networking (CCN) Platform [8].

1.3. Internet of Things

The Internet of Things (IoT) [9] comprises the collection of (smart) 'things' which are connected to the internet. These 'things' can be all kinds of devices capable of sensing and/or changing certain

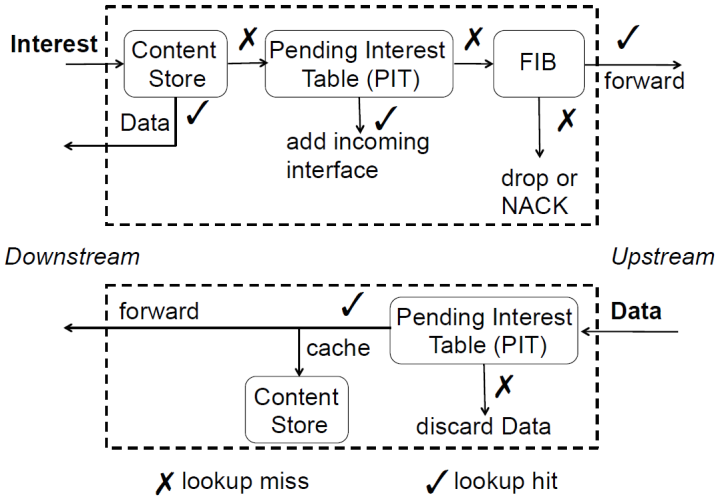


Figure 1.2: The forwarding logic of the ICN architecture. Picture from [4].

phenomena, for example, smart cars, remote temperature sensors, digitally controllable lighting or even smart refrigerators. Connecting these devices to the internet, offers the potential to unlock new types of use cases and applications. For this reason, IoT has risen high market expectations and is envisioned to provide new business opportunities to a large variety of different companies. The IoT is currently at the peak of Gartner's famous hype cycle [10] meaning that it is currently at the highest of its visibility. A lot of different research organisations and consulting firms are studying and making predictions on the future impact of the IoT business and market added value it can deliver. The consulting firm IHS, for example, expects that the number of IoT devices will double in the next five years, reaching 30.7 billion connected devices in 2020 [11]. Management consulting firm McKinsey & Company estimates in a detailed report that the IoT will provide an economical impact of 11.1 trillion USD in 2025 [12].

1.3.1. Key IoT domains

According to McKinsey & Company [12], the large variety of IoT deployments can be structured into nine different IoT settings which are illustrated in Figure 1.3. In the Home and Office settings, IoT devices may be used to monitor and control slow changing climate conditions such as temperature and humidity levels. IoT devices can also help to reduce energy consumption, by using this sensor data to actively adjust heating and lighting settings. IoT devices may also be used for security purposes, by deploying sensors on a large scale to get a detailed overview over the secured home or office.

IoT devices may be deployed in factories, worksites, outside and retail settings to help optimising certain processes. An example can be predictive maintenance for the industrial settings, where IoT sensors can be used to minimise downtime in case of faulty machinery. In retail environments, IoT devices can be used to track customers inside a store and thereby being able to optimise sales. The outside setting covers all IoT use cases that do not fit into the eight other scenarios. Example outside settings are related to navigation at sea, or process optimisation in military operations.

The Human setting focusses on the rise of wearable devices, which mostly perform monitoring actions. These devices are able to measure metrics related to the health of the owner, which can be used for a medical purpose but also for improving the fitness of the user for example with a smart watch.

In the City IoT setting, sensors are employed to more efficiency use the resources in an urban environment. One could use sensors to detect the availability of parking spots, monitor traffic flows and using this data to adjust traffic lights or even dynamically control street lighting by measuring light intensity or road usage.

In vehicular applications the IoT can help to make vehicles autonomous, by combining multiple forms of sensor data gathered from own sensors or other cars. Characteristics of the IoT are currently already visible in modern cars which are able to combine sensor data to propose suitable service intervals. Vehicles operating in harsh conditions may need more frequent maintenance which can be better scheduled if measurement data is used.

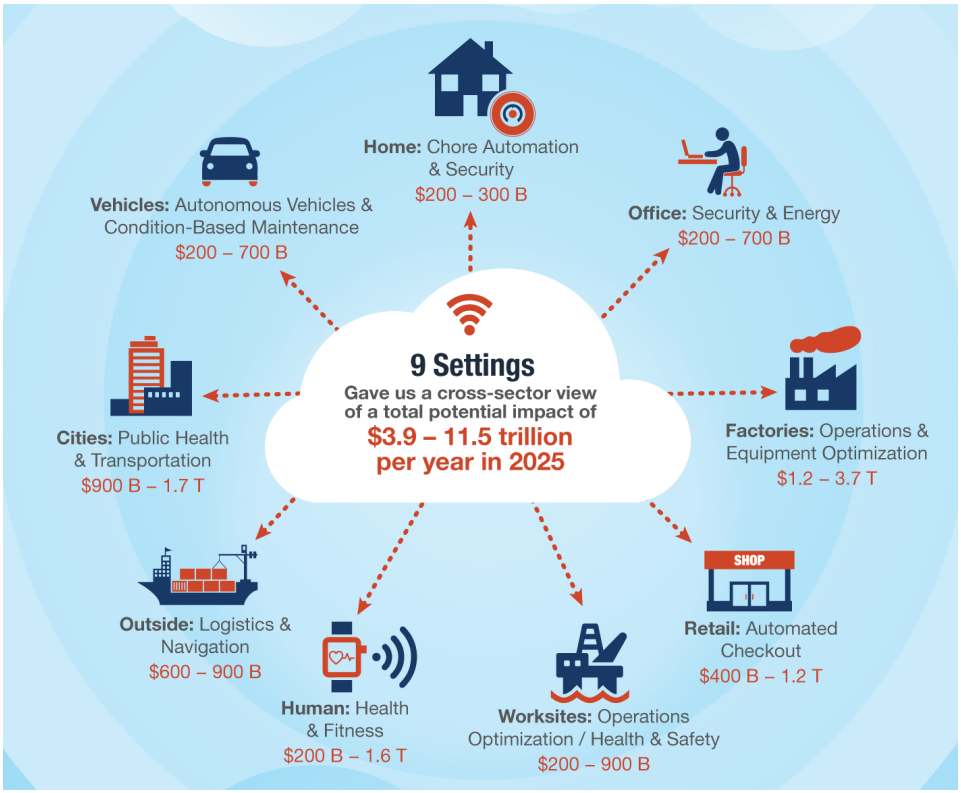


Figure 1.3: The nine most promising IoT settings identified by McKinsey & Comapny. Picture from [12].

1.3.2. Key IoT characteristics

We have identified four key characteristics of IoT networks [13], which we will discuss individually.

- **Sense →Combine →Take Action** Most IoT deployments use a strategy where measurement data from sensors is processed and evaluated to decide whether a certain action must be taken. An energy management set-up in a home setting, may *sense* the temperature in multiple rooms. Motion sensors are able to *sense* where the house owner is currently situated. This data can be cleverly *combined*, which allow the thermostat to dynamically switch off (=*action*) the heating in unused rooms and thereby save energy.

- **Low power, low cost devices** In some IoT deployment scenarios, a power source may not (always) be available. IoT devices then have to be equipped with batteries to supply their needed energy. The devices should be designed in such a way that battery life is maximised, since users do not want to replace batteries often. Battery replacement or charging can become a cumbersome task when the number of deployed IoT nodes increases. Battery life is typically optimised by using constrained hardware with a low performance and therefore a low power consumption. One of the main implementation problems in IoT applications, is to find a balance between application performance and battery life.

- **Large scale distributed data generation** As we stated earlier, it is estimated that the total number of connected IoT devices will increase rapidly in the coming years. This will also cause an increase in generated data traffic. A difference with classical internet traffic patterns is that data is no longer generated at centralised locations such as servers. Data is now generated in a distributed manner over a large number of constrained IoT nodes.

- **Heterogeneous networks** Special physical layer communication technologies exist which allow efficient transmission and reception of data in IoT applications. Examples are 802.15.4 [14], Bluetooth-LE [15], NB-IoT [16], LTE-M [16], LoRA [17], Sigfox [18]. IoT devices may also be connected with classical connection methods, such as wired connections. In the IoT, data from multiple sources

and locations may be exchanged via numerous of different device types and communication methods.

1.4. Qualitative comparison of ICN and IP

A qualitative comparison between ICN and IP helps to determine for which scenarios and use cases ICN may perform better or worse compared to IP. We will use an illustrative example which covers both an IP as well as an ICN enabled network to show the most important differences between both networking technologies. The differences can then be used for a more fine-grained comparison, to find out when these differences may be beneficial in terms of network performance.

1.4.1. ICN vs IP example

In our example which is illustrated in Figure 1.4, two users want to receive a cat picture over both an IP and an ICN enabled network. We have designed the example in such a way that most features of ICN are illustrated. We will first describe how the transmission process of the picture is performed in the IP case, and we will then compare this to the ICN case.

IP case

User C1 (for 'Consumer') wants to receive the picture and it will therefore transmit a request to the server P (for 'Producer') which holds the original cat picture. The user first has to find the IP address of the sever which in this case is 131.180.77.102. C1 creates a request, which is forwarded via node 1, node 2, node 3 and node 4 to the destination server. At every node, the IP routing table is consulted to find the next eligible hop. When the request reaches the server, a response message is constructed, which is transmitted to the IP address of user C1. The response packet is routed via node 4, node 5, node 6 and node 1 back to consumer C1. Just after the transmission of the request from node C1, C2 also issues a request for the picture. The request and response message will follow in this example the same routes as for C1.

ICN case

If user C1 is interested to receive the picture when an ICN network is used, different traffic patterns can be observed. The consumer first has to know the ICN name of the picture, which in this case is '/cat.jpg'. It then creates an Interest message which includes the ICN name of the picture. The consumer consults its FIB to find the next hop for the Interest, which in this case will be node 1. When node 1 has received the Interest it will follow the forwarding logic from Figure 1.2. So it will first check its content store, the PIT and the FIB to forward the data to node 2. The same process will be followed at node 2 and node 3. At node 4, a different path is followed in the forwarding logic since the picture is available from the node's content store. Node 4 retrieves the data from its cache and then creates a

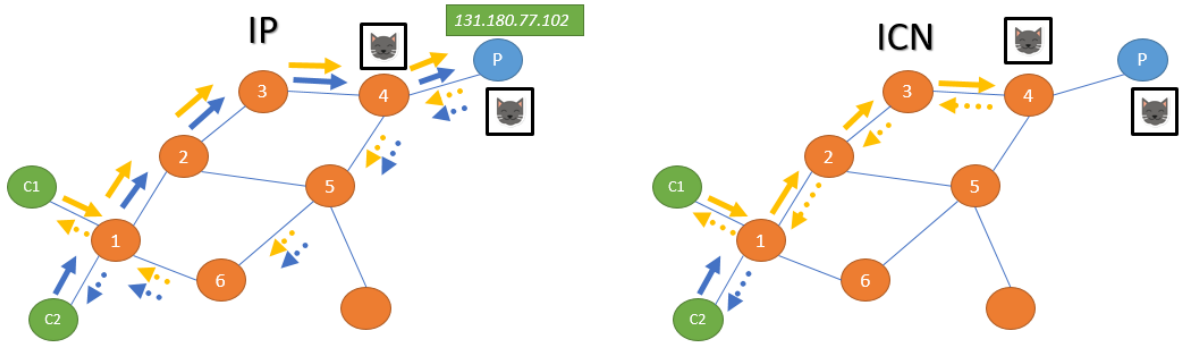


Figure 1.4: The consumers, indicated in green, want to receive a cat picture in an IP (left) and in an ICN (right) network. The producer of the data is marked with a blue circle. The cat picture can be found at the producer node and at node 4, which previously requested the picture. A solid arrow indicates the transmission of a request/Interest message, while a dotted arrow indicates the transmission of a response/Data message.

Data message which is forwarded back to the consumer by consulting the PITs for every node on the return path. The data then arrives at node C1.

Just as was the case in IP, C2 decides to also issue an Interest for the picture just after C1's transmission. An Interest is created which is forwarded to node 1. The Interest enters the forwarding logic at node 1, where the PIT entry from the C1 Interest matches C2's Interest. An extra incoming face will be added to the PIT entry, and the Interest will not be forwarded further. The Data message which satisfies the Interest from C1 will also be forwarded to C2, and thus also satisfying C2's Interest.

In this specific example, the IP network required 20 message transmissions to satisfy the requests from node 1 and node 2. On the other hand, in the ICN network only 10 message transmissions where needed to satisfy the interests.

1.4.2. Main differences between ICN and IP

The example in Section 1.4.1 shows that there are large differences in the operational principles of IP and ICN. We identified the five most important differences between both networking technologies.

- **Data naming vs. IP addresses** This is the most trivial difference between ICN and IP. Since ICN addresses data, names are used to specify which content a node wants to retrieve. In our example, C1 and C2 had to first find the correct name and IP address to properly address the cat picture. One of the advantages of ICN, which is commonly stated by the ICN community, is the elimination of the necessary translation between the content name and the right IP address. Since ICN is using the data name itself one could directly request the appropriate data. This is however not always true, since a large scale deployment of ICN would also require a translation step between the name of the content and ICN names which are known in the ICN network. So for both ICN and IP a name resolution step will be necessary in most cases, either to find the IP address, or to find the data name.

  One of the advantages of using IP addresses is that the size is typically fixed. An IPv4 address will always be 4 bytes long and IPv6 addresses will take 16 bytes of space. ICN Interest and Data messages always have to contain the data name, which can be of variable size. The exact size of the data name may depend on the specific application and naming scheme used.

- **Message types** As discussed earlier, ICN uses two fixed message types: the Interest and Data message. This is one of the core features of ICN to allow the deployment of advanced forwarding and distributed caching. IP does not set any constraints on message types and all IP packets may be routed the same way. Compared to the CCNx and NDNx ICN implementations, IP is easier to process since it is using fixed headers. The ICN implementations currently use so-called type-length-value (TLV) formatting, which allows to easily add extra header fields at the cost of having a more resource-intensive processing step in forwarding devices.

- **Stateful vs. stateless forwarding** ICN uses stateful forwarding, while IP uses stateless forwarding. ICN thus stores information about earlier transmissions in a complex forwarding pipeline with PITs and CSs. IP is considered stateless since forwarding decisions are typically not based on earlier decisions. This difference is clearly illustrated in the example in Figure 1.4, which shows that in the IP case the routing paths of the request and response message are not necessarily the same and in this case are not completely overlapping. This is different in ICN since the Data message always has to follow the path of the Interest message, which requires a stateful forwarding scheme. Moreover, the stateful forwarding in ICN allows the use of Interest aggregation, which is also illustrated in Figure 1.4, a second Interest for the same data entering an ICN node will typically not result in the forwarding of that Interest. Only the incoming face is registered in the PIT of the specific intermediate node. Compared to IP, this can greatly reduce the required network bandwidth usage since the Data message would only be transmitted once. Only at node 1 in our example, the Data message will be transmitted two times, but over two different interfaces.

  Depending on the specific traffic patterns, this more advanced forwarding logic of ICN may lead to a more efficient network usage. Naturally, the more nodes are interested in the same data at around the same time, the more bandwidth may be saved. Also the topology may have a big impact on the possible performance increase caused by this stateful forwarding plane. If most nodes have node-disjoint paths to the producer, the benefits of using interest and data aggregation may be small.

- **Application layer vs. network layer caching** The distinction between Interest and Data messages in ICN, together with the use of named data, creates a possibility to easily deploy caching at the network layer on a network-wide scale. In IP, caching is also possible but it is harder to deploy on the network layer. This because IP packets do indicate the origin and destination address, but do not typically indicate what data is actually transported. This makes it hard to recognise duplicate requests for the same data. Therefore caching is typically deployed at layers above the network layer, which do have information on the data which is transmitted. IP caching implementations are therefore mostly application-specific and require more configuration effort than is the case for ICN. The usefulness of having caching in all nodes may also be dependent on the deployment and use cases. A network with distributed content generation and retrieval may benefit more from a distributed caching approach compared to classical hierarchical networks where the deployment process of centralised caches is more straightforward.

- **Legacy deployment and interoperability** One of the design properties of ICN is that it can run over any other communication protocol or technology. ICN can be deployed as a replacement of IP but also on top of IP. The current internet is using mainly IP and network-wide adaptation of ICN as a complete replacement of IP would require a large effort and hence be very costly. The advantage of using ICN on top of IP, for example in UDP packets, is that it would allow a straightforward deployment on the current internet. The disadvantage of this approach is that the resulting data overhead from the UDP and IP headers may cause a higher network usage, compared to the case when ICN is deployed at the network layer. There is therefore a trade-off between having a more advanced protocol with stateful forwarding and caching which may lower the network usage, and the extra overhead caused by the deployment on the legacy internet.

1.5. Research objective: Applying ICN to IoT

According to its creators, ICN is designed to fit the way we use the internet better than IP currently does. The use of named data and distributed network layer caching may provide a more efficient utilization of network resources due to the stateful forwarding plane which allows data to be retrieved from caches close by the requester, while also providing a higher content delivery performance in terms of content retrieval delay. Since the IoT is expected to connect billions of devices to the internet, a resource-efficient network paradigm is needed to cope with the corresponding enormous traffic increase. IoT deployments also typically follow a distributed data generation and retrieval paradigm, which could benefit from ICN's in-network caching approach and stateful forwarding logic. This thesis focuses on assessing whether ICN is advantageous for the IoT in these aspects, by comparing an ICN approach to an IP approach for IoT applications. In particular, this thesis addresses the following main research question, which is supported by three sub-research questions.

- Which technology performs better according to predefined metrics, to what degree and in which scenarios?
  - Which architectures for both the IP and ICN approach should be used?
  - Which relevant IoT scenarios should be defined for a fair comparison between ICN and IP?
  - Which metrics are relevant for the targeted comparison in IoT scenarios?

1.6. Thesis outline

This thesis is structured as follows. Chapter 2 describes the relevant prior art in the context of ICN for IoT and describes how this thesis contributes to this line of research. In Chapter 3 the selected scenarios are introduced, including the method used to make the selection. In Chapter 4, the performance evaluation methodology is outlined, along with the implementation model that was needed for carrying out the evaluation. Chapter 5 presents and analyses the results of the performance evaluation and Chapter 6 concludes this thesis with a summary of the findings and a sketch of future research directions.

2

Literature review

This chapter presents a thorough review of the state of the art in the research field of applying ICN to IoT. The goal of this effort is to discover the research themes covered by previous work and identify promising areas where further work is required and where a meaningful contribution can be made. Section 2.1, introduces these research themes and describes them by discussing the most relevant articles. Based on the analysis done in Section 2.1, the chosen theme for this thesis is presented in Section 2.2, along with its contribution with respect to previous work.

2.1. Research done in the ICN for IoT field

We have identified 65 scientific publications which cover subjects related to the ICN for IoT research field. From an analysis of these papers we have derived seven categories, or *themes*, for the ICN research on IoT. Figure 2.1 below provides a visual representation of these themes and the work done in each of them by means of color-coding based on the amount of scientific publications in each respective theme. A sheet which shows the mapping of these 65 papers to the different categories is included in Appendix A.

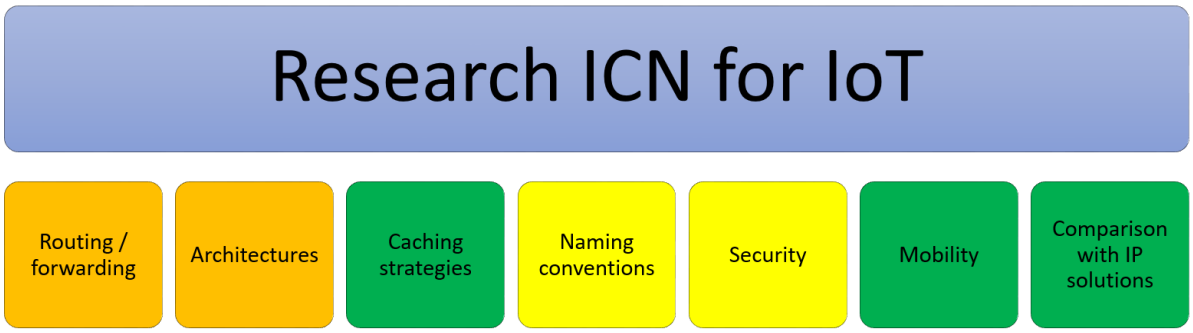


Figure 2.1: Identified research themes in the ICN for IoT field. The category colour defines the number of papers which cover this specific subject, green: 0-5 papers, yellow: 6-10 papers and orange: >10 papers.

In the following, we will describe the focus of the research done within each category and highlight the papers providing the most significant contributions.

2.1.1. Routing and forwarding

ICN deployments require special routing and forwarding algorithms which are able to find the most suitable next hops for each Interest message. Instead of finding paths to specific hosts such as in IP, ICN routing protocols need to find the best route to one or multiple producers/caches for each data name. Research on routing and forwarding in the ICN for IoT field mainly focuses on developing energy-efficient routing and forwarding schemes which can be used on constrained IoT devices. As can be

concluded from Figure 2.1, this is one of the two most popular research themes. Authors typically focus on finding the most energy-efficient balance between signalling traffic and the use of traffic flooding, mostly for mesh type networks. The work done in this category may be classified into blind forwarding and aware forwarding [19]. Blind forwarding algorithms typically rely on computationally inexpensive (controlled) flooding of data over a network, with limited signalling overhead. With aware forwarding, routing and forwarding strategies aim at gathering higher level information about, for example, the network topology, and use this information to implement efficient forwarding, typically at the cost of having more signalling overhead.

Examples of aware forwarding strategy proposals are [20] and [21] . In [20] the authors propose a technology called GRMR: Greedy Regional Multicast Routing, which uses local multicast tree constructions to find the most efficient routes. In [21] the authors present a hybrid combination of aware and blind forwarding. In [22] two blind forwarding strategies are proposed, Vanilla Interest Flooding (VIF) which floods interests over the network and Reactive Optimistic Name-based Routing (RONR) which creates FIB entries after a single interest flooding. An adaptive forwarding scheme is developed by [23], which lets IoT nodes with the highest battery charge perform the flooding operation.

2.1.2. Architectures

As can be seen from Figure 2.1, a large fraction of the papers covering ICN for IoT research study the development of new architectures. The researchers aim to design mechanisms to apply and adapt the ICN paradigm to a certain, typically already existing, scenario or technology. Most of the times, these architectures are roughly based on the principles of information centric networking defined by Van Jacobson [3]. However, in some cases, the authors propose fundamental changes to the ICN paradigm, such as eliminating ICN's one-Interest one-Data policy, to support a specific use case. In these cases the authors typically focus on high-level design rather than thorough experimentation with the newly proposed architecture.

Among the papers defining new architectures [24–30], the most relevant to our research questions are [28] and [29]. In [28] an architecture is developed to support the deployment of ICN for the IoT in 5G cellular networks. The authors develop a controlled Shared Caching System (SCS) for Fixed and Mobile Converged (FMC) networks. Another interesting proposal is [29], where architectures combining ICN, SDN and IoT are introduced. Multiple architectures are analysed which combine ICN and SDN into Sensing As A Service (SAAS) [30] cloud paradigms.

2.1.3. Caching strategies

A less popular research theme is the development of optimised caching strategies for the IoT. The design of caching strategies may not be seen as an ICN-specific research challenge, since caching is already studied and adopted in a wide range of applications. Caching stategies for IoT applications can be optimised for specific IoT aspects, such as traffic patterns and device constraints. We can identify two main aspects in caching strategies, namely cache decision strategies and cache replacement strategies. A *cache decision* strategy decides whether incoming Data messages need to be cached. This decision may be taken based on predefined criteria, for example, randomly, based on the popularity of a certain Data message or on the current contents of the cache. A probabilistic approach is shown in [23], which allows caching of incoming data to be decided based on a random variable. The presented approach is compared with the common 'cache everything' strategy. A more advanced caching policy is proposed in [31], where the decision to cache a certain piece of data is based on the weighted sum of the battery life, cache occupancy, and the remaining time until the Data is considered stale.

The second aspect of caching strategies is the *cache replacement* strategies. These policies come into action when a cache reaches its full occupancy. When a new Data message arrives and if the cache decision strategy decides to cache that particular data, then this packet may need to take the place of a previously cached one. The cache replacement strategy has to decide which Data message to replace, based on some predefined logic. An example of a cache replacement strategy for IoT applications called Max Diversity Most Recent (MDMR) is proposed by [32]. The MDMR strategy aims to maximise the availability of data from multiple producers in a single cache. When a new Data message arrives, the oldest Data from the same producer is replaced. If there is no data from this producer in cache, then the oldest Data from another producer will be replaced.

2.1.4. Naming conventions

How Data messages should be named is still an open challenge for ICN applications in general and for IoT applications in particular. Typically, authors acknowledge the design problem of naming conventions, but still leave it as future work. In [33] some important research challenges are described for defining names in IoT applications, such as: “How to deal with the typically long names in IoT applications?”, “Should we name data based on metadata?” and “How should we deal with dynamic data that is changing over time?”. Many of these challenges have still not been addressed in ICN for IoT proposals.

2.1.5. Security

The third most popular research theme is security. As explained in Section 1.2.1, ICN does not use secured connections such as in IP, but uses content-based security where individual Data messages can be encrypted and self-verified. This requires the development of new security technologies and protocols. The use of robust security technologies is especially important in IoT applications since IoT devices may gather sensitive private information.

Nine, from the total of 65 papers, focused on security aspects of ICN for IoT applications. A broad spectrum of security issues have been addressed in literature ranging from authentication and authorisation processes of new ICN-enabled IoT nodes [34, 35], to complete NDN security architectures [36, 37]. In [34] the ‘OnboardICNg’ authentication and authorisation protocol is presented which uses a central authentication server. The authors of [35] compare the ‘OnboardICNg’ protocol with an authentication protocol using asymmetric encryption, which eliminates the need for having a central authentication server. It is shown that this advantage comes at the cost of a higher latency and energy usage due to the more resource-intensive cryptographic operations.

In [38] Attribute-Based Encryption (ABE) for IoT applications is described, which allows to make cached encrypted Data messages available to multiple users. When a Data message is encrypted at the Producer, ABE can be used to specify certain users (with certain attributes) which must be able to decrypt the message from the sensor node. This technology allows multiple consumers to retrieve and use encrypted Data messages from network caches.

2.1.6. Mobility

Several IoT use cases include mobile nodes. For instance, a tracking device may switch multiple times between different connection points when its tracking the location of some asset. ICN natively supports consumer mobility. A consumer can just retransmit its interest when it connects to a new point of attachment and continue to receive and request data along the new path. Producer mobility is not natively supported by the ICN architecture. If a producer moves, routing entries in FIBs need to be updated. ICN nodes will otherwise forward interests to old producer locations which are still present in their FIBs. Producer mobility support is a fundamental challenge of the ICN architecture and is therefore not specific for IoT applications. In survey paper [39] an overview is given of multiple proposed solutions. The authors identify four types of solutions to support producer mobility in ICN.

- **Mapping-based solutions** These solutions use a so-called rendezvous (RV), which keeps track of the current location of the producer. When a producer moves to a different location it has to report its new point of attachment to the RV. A consumer then may consult the RV to retrieve the new location/name of the mobile producer.

- **Tracing-based solutions** Tracing-based solutions also use an RV and additionally benefit from NDN’s stateful forwarding plane to update the producers location. Interests for the data of the mobile producer are forwarded to the RV. The mobile producer regularly sends special *trace command interest* messages to the RV to retrieve these pending Interests.

- **Data spot** In some specific applications a data spot approach can be used. This technique uses ICN names which are coupled to geographical locations. An example can be */RWS/a4/hmp45_4/Temp*, which requests the current temperature at a specific location along highway A4. Any node that is currently at this location may satisfy this request.

- **Data depot** This mobility technique uses a central depot which stores all data of mobile producers. Mobile producers will send all generated data to this fixed depot node. All Interests for mobile producer data messages are satisfied by the data depot.

Although producer mobility is not an IoT-specific problem, efficient mobility solutions should be developed for resource-constrained nodes. Producer mobility support for IoT applications is still a relatively unexplored research theme.

2.1.7. Comparison with IP solutions

Only two of the 65 covered papers attempt to perform a comparison between ICN and IP for IoT use cases, to find out whether ICN really improves network performance in IoT applications [22, 40]. In [40] a very simplistic topology is used for the comparison and NDN is compared with an IP-based protocol stack which is not optimized for IoT applications. In [22] NDN is compared with a representative IoT-optimised IP stack, but only a single artificial use case is covered. Both papers lack any descriptions about the used traffic patterns and popularity distribution. Moreover, both papers cover only cover one single use case. Most importantly, no scenario is considered where multiple IoT deployments are interconnected via the realistic internet-like topologies. Therefore it can be concluded that a thorough comparison of ICN and IP for IoT applications is still missing in literature.

2.2. Focus of this thesis

The literature review showed that comparing ICN and IP for IoT is still a largely unexplored field. Yet, understanding how ICN performs compared to IP is a very important aspect to tackle, if ICN ever wants to take a dominant role as network protocol for the IoT. In this thesis we make a first important step into this direction by benchmarking ICN's performance against that achieved by traditional IP in a range of relevant IoT scenarios, because it can be seen as the first and foremost key unanswered research question regarding the usage of ICN for the IoT. In fact, any follow-up research on, for example, mobility support in IoT applications may only be relevant if the ICN paradigm is proven to be worthwhile.

3

Comparison methodology

Comparing the performance of ICN and IP for IoT applications requires the definition of several scenarios. The scenarios will be based on realistic IoT use cases and use state of the art implementations of ICN and IP for IoT. In this way, a representative comparison can be ensured. This chapter will describe the process of selecting and defining these relevant scenarios.

Section 3.1 introduces the method used for selecting IoT use cases, which will serve as a basis in Section 3.2 to create the comparison scenarios. Section 3.2 will also describe the process of selecting appropriate metrics which will be used for our comparison.

3.1. IoT Use Cases

As mentioned in Chapter 1, the Internet of Things is a popular topic in scientific literature with currently already over 27,000 contributions. Nevertheless, we could hardly find any research providing a qualitative overview of different IoT use cases and applications. One of the studies which does provide such an overview is the McKinsey report on IoT [12]. In this work, over 300 IoT use cases were mapped onto nine different categories, as previously described in Section 1.3.1.

The authors estimate the economic potential for each category by using economic modelling techniques. We use this document to select two promising use cases to be used for our comparison. According to the McKinsey report, the IoT will provide a potential economic impact of 3.9 trillion to 11.1 trillion USD in the year 2025. Two-thirds of the economical value created by IoT deployments will originate in business-to-business (B2B) situations. So most impact will be concentrated around IoT applications in commercial environments such as factories, work sites and production locations. The remaining one-third of the value resides in the consumer market which comprises, for example, wearable devices, health care and 'smart home' products.

Based on the insights from the McKinsey report, for our comparison we choose one consumer use case and one business use case. Additionally, to ensure that we cover a wide spectrum of scenario aspects, to cater for a large part of possible IoT applications. We therefore choose the scenarios in such a way that extensive sensitivity studies for both scenarios cover a wide range of IoT deployments.

From the consumer market use cases we choose the 'smart home' use case since it has a big estimated impact [12]. This scenario also allows us to cover monitoring use cases, which can be found in several IoT settings such as the city, vehicles, wearables and office settings from [12]. In a *smart home*, all kinds of ordinary appliances are connected to each other and to the internet. Smart thermostats can use temperature information from sensors in different rooms to only warm up certain low temperature rooms, and in that way save energy. Lights can be dynamically switched on and off based on the location of the home owner. This can be done with, for example, motion sensors which also can be used for security purposes. Following the same reasoning for the B2B use cases, we select the *smart factory* use case based on the estimated impact [12]. Moreover, this scenario allows us to cover process optimisation applications which are also present in many IoT applications such as the worksites setting, outside setting and retail setting from [12]. The *smart factory* use case focuses on optimising operations by continuously monitoring parts of the production process, and to use this information to actively perform adjustments. In [12], a real-life example is presented where humidity

sensors are used to optimise the quality of paint jobs at the production facility of General Motors. Car parts are routed to the best possible location based on this sensor data.

3.2. Comparison approach

This section will present the comparison approach, which is used to define the comparison scenarios representing the use cases from Section 3.1. In Section 3.2.1 we create a scenario framework by identifying relevant scenario aspects. Next, in Section 3.2.2 we select the network technologies used in our comparison. The scenario framework is then used in Section 3.2.3 to define all scenario aspects for the 'smart home' and 'smart factory' scenarios. Finally in Section 3.2.4 we introduce the Key Performance Indicators (KPIs) or metrics used.

3.2.1. Scenario aspects

Each comparison scenario is composed out of multiple aspects, for example, the number of IoT nodes, their topology or the frequency of issued requests. We will identify multiple relevant scenarios aspects to build a scenario framework. We divided the parameters into three different groups, namely topology related aspects, traffic patterns and node characteristics. An overview of the identified scenario aspects can be found in Table 3.1.

Topology

We want to compare ICN and IP for IoT using a realistic network topology. We therefore follow the approach of having multiple IoT islands each connecting multiple IoT nodes. These IoT islands are connected to the internet via special gateway routers. The gateway routers are able to communicate with both a conventional network and the IoT nodes which typically use a special radio access technology and network stack. The IoT nodes are also able to communicate with other IoT nodes in the same island directly. In a 'smart home' scenario an IoT island may represent one home, while in a 'smart factory' scenario an IoT island may only cover a small part of a factory hall.

IoT islands

Every IoT island comprises a fixed number of IoT nodes per scenario. We call this scenario aspect: *Number of IoT nodes per island*. The second parameter which has influence over the number of nodes, is the *number of IoT islands*, and it defines the total number of IoT islands in a scenario. Each of the specific scenarios considered will have different fixed values for these parameters. In Figure 3.1 an illustrative example is shown of an IoT network where the *Number of sensors per island* is set to twelve and the *number of IoT islands* is set to six.

Backhaul network

The IoT islands are connected to other islands via their gateway router. We call the network which interconnects multiple IoT islands the backhaul network. For each scenario a different backhaul network

Table 3.1: Overview of identified scenario aspects.

Topology	Number of IoT islands
	Number of sensors per island
	Number of backhaul nodes
Traffic patterns	Leaf node consumers (CL1)
	Gateway consumers (CL2)
	Inside island consumers (CL3)
	Backhaul network
	Data freshness period
	Request frequency
	Content popularity distribution
	Data payload size (bytes)
Node characteristics	Cache size (kB/packets)

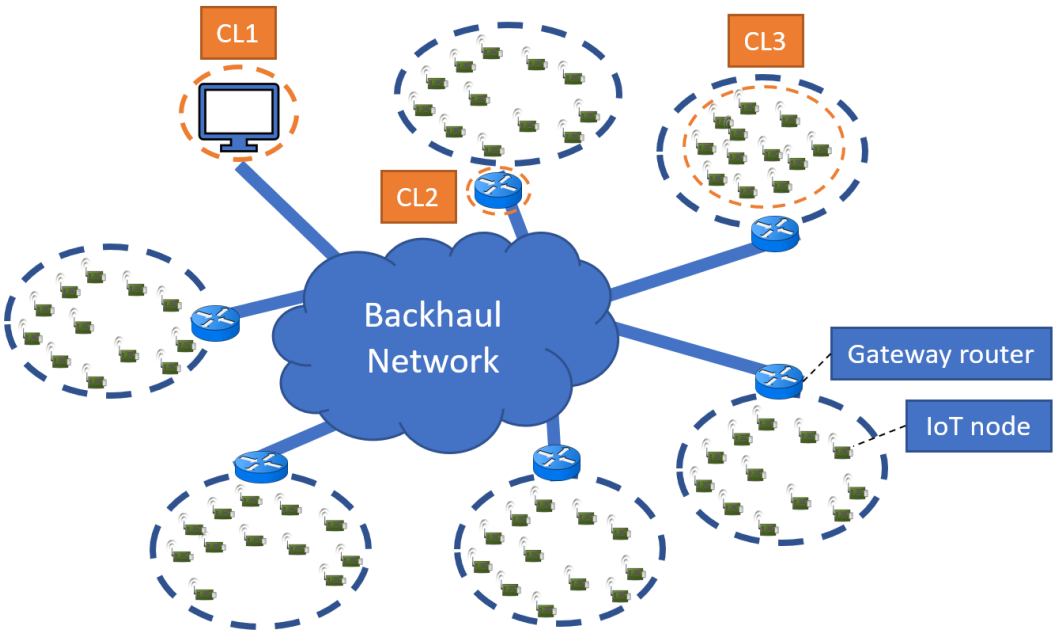


Figure 3.1: IoT islands scenario. The picture also illustrates the three different types of consumer locations (CL1, CL2, CL3)

may be chosen. For example, we may want to model a small corporate network for the 'smart factory' case or an internet-like topology for the 'smart home' scenario. We model the backhaul networks via random graphs. The IoT islands are connected to a randomly chosen terminal node in the random graph.

Traffic patterns

Specifying the traffic patterns is one of the most crucial aspects in the design of the comparison scenarios. This parameter has a big impact on the results of the comparative study. We borrow the words *consumer* and *producer* from the ICN paradigm, and we use them in both the ICN and IP scenarios. The node which requests data is called a *consumer* and the node which creates new data is called a *producer*.

Consumer/producer location

Based on the selected IoT use cases, we consider three different types of consumer locations, which we call CL1, CL2 and CL3.

Consumers with location type CL1 are located at randomly selected leaf nodes in the backhaul network. These consumers may represent hosts, servers or users outside the IoT islands interested in sensor data. The second location type is CL2, where consumers reside on gateway nodes. The last consumer location type: CL3 covers IoT sensors nodes inside the IoT island. This location is relevant when IoT nodes, which may already have a producer role, are also interested in data from other IoT nodes. The three types of consumer locations are also shown graphically in Figure 3.1.

IoT sensor nodes are the only nodes in the topology with a producer role, the producers will thus always reside in the IoT islands.

Data freshness period

The data freshness period is an important parameter for all scenarios where caches are deployed. It is a setting for the cache of a node and configures the time until cached data is considered stale. If the data freshness period of data is zero, then nodes are not allowed to cache this data. If the data freshness period is set to, for example, one second, all caches are allowed to satisfy requests/Interests with a cached copy of this data for a duration of one second.

Request frequency
Consumers request data from IoT nodes at specific points in time. For simplicity, we assign a fixed request frequency to every consumer. A consumer starts to send requests at this frequency after a randomly chosen initialisation delay. This delayed start prevents that all nodes send requests at exactly the same time. The requests are issued independently of a successful reception of application data.

Content popularity distribution
Not all data produced by IoT sensors will be of equal importance for data consumers. Some sensors could provide information which is more popular, resulting in a higher number of consumer requests for this data. To model this difference in popularity between producer nodes, we define a content popularity distribution for each comparison scenario. The distribution will be used by the consumer nodes to decide what data item to request at the fixed intervals defined by the request frequency.

Data payload size
We define the data payload size to be the number of bytes of the raw data requested by the application layer for both ICN and IP. This data payload size will be defined for every scenario.

Backhaul network IP/ICN
As can be understood, for each scenario, we deploy an IP as well as an ICN network stack to compare network performance. Additionally, we also cover a hybrid option where ICN-enabled IoT islands are connected to an IP-based backhaul network. This is a relevant deployment case since it is expected that ICN will be introduced gradually and will need to be interconnected with the current internet backbone which uses IP [41].

Node characteristics
The last scenario aspect group comprises all parameters related to node characteristics which may have impact on the performance of IoT networks.

Cache capability
An advantage of the fact that ICN packets address data rather than hosts is that ICN data packets can be cached at any place within the network, in order to more efficiently serve future data requests too. Caches can be deployed on all network nodes which have sufficient storage available. For the IP case we will also consider scenarios where caching is used, to allow for a fair comparison between ICN and IP. Caching in IP networks is only possible if specialised higher layer protocols are used. We will select this IP based communication technology in Section 3.2.2.

Cache size
The cache size for the ICN case is defined in terms of data packets that it can store. This value should be adapted to the IoT use case based on the device constraints of the IoT node. For the IP case we will use an equal size cache.

3.2.2. Network model
The selection and definition of the used networking technologies is done using a systematic approach which follows the layers of the OSI-model. We start off by selecting relevant technologies at the physical layer and continue towards the higher layers, until the application layer. For each of the OSI layers we select a network technology fit for the IP case and one fit for the ICN case. To enable a fair comparison, we select, whenever possible, the same or a comparable technology for the IP case and the ICN case. An overview of the OSI layers and selected technologies for IP and ICN is reported in Figure 3.2. A detailed discussion of the choices made is given below.

L1: Physical layer + L2: Data link layer
For the first two OSI layers we use the same protocols for the ICN and IP cases. Specifically, on the physical layer we use the IEEE 802.15.4 wireless standard [14] to model low power IoT nodes. The protocol, which has been designed for use in low-rate wireless personal area networks (LR-WPAN), is

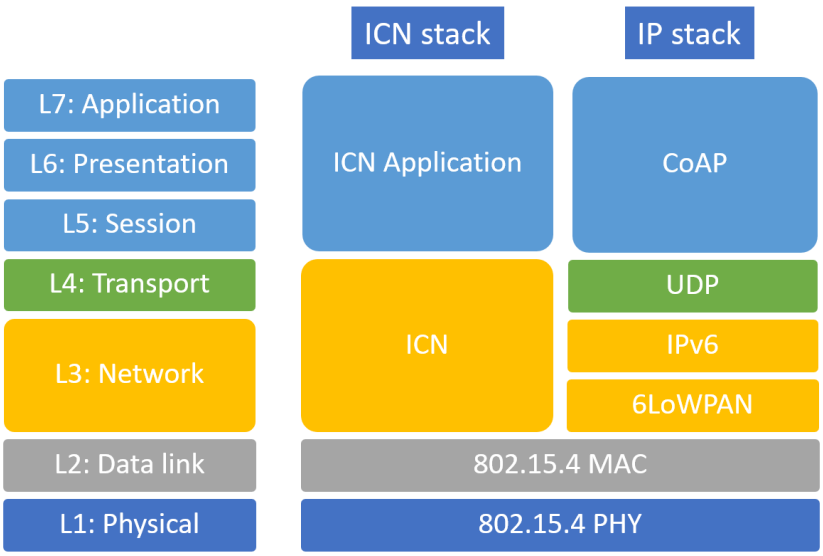


Figure 3.2: Selected network stacks for ICN and IP.

currently used in many IoT network standards such as the Thread network protocol [42], WirelessHART communication protocol [43] and the well-known ZigBee specification [44]. The IEEE 802.15.4 standard allows low complexity data reception and transmission at low energy usage making it suitable for constrained IoT devices. On the data link layer we use the default MAC defined in the 802.15.4 standard.

L3: Network layer

IP

On the network layer we use IPv6 with the 6LoWPAN adaptation layer [45]. This adaptation layer is needed to allow efficient transmission of IPv6 packets over constrained LR-WPAN links. The IPv6 standard [46] requires the data link layer to be able to transmit IPv6 packets with a size of 1280 octets in one piece, thus without fragmentation. This requires the 802.15.4 MAC layer to transmit frames with a payload size of 1280 bytes, which is impossible since the 802.15.4 physical layer has a maximum transmission unit (MTU) of 127 bytes. Therefore the IETF standardised a special adaptation layer placed between L2 and L3 of the OSI model, to allow direct end-to-end IPv6 traffic with LR-WPAN nodes. When an IPv6 packet arrives at a 6LoWPAN enabled IoT island, then the gateway node will replace the large IPv6 header with a smaller and more efficient 6LoWPAN header, before transmitting the packet on the 802.15.4 wireless link. The opposite process is executed when packets are sent from IoT sensors to outside the IoT islands.

ICN

In Section 1.2.2 we have introduced the two main competing ICN implementations from the Content Centric Networking (CCN) architecture [7] and the Named Data Networking (NDN) architecture [6]. For this comparison we use the NDN architecture design since it is currently the leading ICN architecture in literature, and the architecture consequently has the most mature software implementations and evaluation platforms available.

L4: Transport

IP

When modeling resource-constrained IoT networks, UDP is typically favoured over TCP [22] due to the lower overhead. The connectionless nature of UDP also enables a more fair comparison with the equally connectionless approach of ICN.

ICN

ICN follows a different architecture without a separate transport layer, as is depicted in Figure 3.2.

We therefore do not have to select a L4 network protocol for the ICN scenarios. Moreover, since ICN addresses names rather than hosts and ports, and given the choice of UDP for the IP case, no additional transport layer is needed in the ICN case.

L5: Session layer + L6: Presentation + L7: Application layer

IP

Several application layer protocols exist which are suitable for IP-based IoT applications, such as the Extensible Messaging and Presence Protocol (XMPP) [47], the Message Queuing Telemetry Transport (MQTT) protocol [48], the Advanced Message Queuing Protocol (AMQP) [49], the HTTP Representational state transfer (REST) services [50] and the Constrained Application Protocol (CoAP) [51]. The authors of [52] provide a comprehensive overview of the benefits and drawbacks of these application layer protocols. CoAP, XMPP and HTTP REST services use a request-response architecture, where a user can acquire data with custom request messages. MQTT, AMQP and XMPP use a more advanced publish-subscribe patterns, where users can subscribe for specific content.

To ensure a fair comparison between ICN and IP, we have to select application level protocols with similar behaviours. Since the NDN architecture does not yet natively support push-based publish-subscribe traffic patterns, even though efforts currently take place [53] to add this functionality, we therefore select request-response based protocols for the IP case as well. From this subset of protocols we opted for CoAP, since it is the most modern IoT-optimised application layer standardised by the IETF [51].

The CoAP protocol follows the REST model, like HTTP, with support for GET, PUT, POST and DELETE methods. Users can request data by sending a GET message containing the Uniform Resource Identifier (URI) of the data, such as GET/humidity, to an IoT node. The IoT node will then respond with a CoAP packet containing the data. The CoAP protocol also allows to use caching of CoAP messages at dedicated caching nodes. For the scenarios in the IP case where caching is used, we will enable CoAP caching at all gateway routers.

For the sake of comparison between IP and ICN, we assume that consumers know the mapping from a specific data producer to an IP address and we do not consider the traffic that would be introduced if DNS services would be used.

ICN

Since the NDN architecture is already built following the request-response model, no additional layer needs to be introduced below the application layer. Rather, we built a custom application that directly issues Interest messages for IoT sensor data. We therefore build a custom application based on the example scripts provided by the NDN implementation [6], that directly issues interest messages for IoT sensor data.

Similarly to the IP case, where we assumed consumers to know the mapping between IP address and data producer, for the ICN case we assume that consumers know the name of the data they want to receive.

3.2.3. Scenario definition

In this section we will use the comparison scenario framework from Section 3.2.1, to define realistic parameter settings for each of the selected scenarios from Section 3.1. Additionally, we will outline for each scenario how the differences between ICN and IP, identified in Section 1.4.2, may be be reflected in the covered scenario.

‘Smart home’: baseline scenario and sensitivity analysis

For the ‘smart home’ use case we define a baseline scenario and a number of scenarios meant for sensitivity study, where, in each scenario, one of the parameters of the baseline scenario is varied, to test its effect on both the IP and ICN cases. The baseline and sensitivity study scenarios comprise six subscenarios (S1-S6 in Table 3.2), which cover all combinations of backhaul networks and cache configurations. For the IP case, two subscenarios can then be defined: one scenario where caching is disabled and one scenario where caching is enabled. The ICN case will contain four subscenarios, since here we also need to cover combinations where ICN is used with an IP backhaul network, as described in Section 3.2.1. Table 3.2 shows the used parameter settings for both the baseline scenario

and the sensitivity analysis, which we will discuss further in this section. The *sensitivity study* column should be read as follows: each entry in the column typically represents one unilateral variation with respect to the baseline scenario, with all other aspects are kept the same. The sensitivity study for the content popularity distribution is an exception to this rule, since we here also use a smaller cache size to be able to show the differences between the chosen parameter settings.

Topology

We will now discuss the chosen parameter settings for the baseline case, following the order used in Table 3.2. Our goal is to make the evaluation as realistic as possible. Therefore we do not put constraints on the number of the IoT islands and the number of the backhaul nodes to be considered. Obviously, an upper bound will be derived from the technical possibilities of the evaluation platform that we will choose. This number depends on the constraints imposed by the evaluation platform we selected (see Section 4.1.2). For the smart home scenario, our platform is able to support a maximum of 32 islands, connected to a backhaul network with 400 nodes.

Regarding the number of IoT nodes per Island, our goal is to obtain the "correct" order of magnitude. To achieve this, we consider a 'smart home' to have five rooms with each with about five sensors. For example, three nodes for controlling lights and two sensors (e.g. temperature, humidity) per room. This leads to a total of 25 IoT nodes per home.

Traffic patterns

We assume there will be two external consumers (CL1) interested in each 'smart home' sensor data, representing remote home owners or servers, and two internal consumers (CL2), to model local data consumers present in the 'smart home' itself. We set the freshness period to five minutes for the 'smart home' case, following the reasoning that data in 'smart home' cases such as rooms temperatures are not considered extremely volatile, since for example, temperature does not change drastically in a five minute period and longer freshness periods may be allowed.

We then have to define the content popularity distribution for both the IP and NDN scenarios. We have spent a considerable amount of effort researching appropriate popularity distributions for IoT applications. We found that most literature on IoT traffic modelling is focussed on the arrival process of IoT traffic [54–56], and not necessarily on the distribution of requests for certain content which we are after. Following the suggestions of the IETF document [57] regarding ICN evaluation methodologies, we selected a popularity distribution typical for web-based internet traffic, which follows a Zipf distribution. The Zipf distribution makes use of a ranked list of data elements. The data with the highest popularity has the highest rank $i = 1$, the data with the second-highest popularity has rank i = 2, etc. The probability that a node requests data with a rank= i, then equals:

$$p(i) = \frac{K}{i^{\alpha}} \tag{3.1}$$

where

$K = \sum_{i=1}^{N} \frac{1}{i^{\alpha}}$ is the Normalisation parameter
$N =$ Total number of content chunks
$\alpha =$ Distribution parameter

According to [57], realistic values for α lie between 0.64 and 0.84 for web-based applications. We set $\alpha = 0.64$ and use the sensitivity analysis to see the effect of this parameter on the performance of both IP and ICN networking paradigms.

For simplicity, we use a fixed request frequency for requests submitted by data consumers. We assume that a consumer may want to know the room temperatures on an hourly basis, resulting in an order of ten requests per hour. Every consumer is therefore assigned a fixed request frequency of ten requests per hour. As mentioned in Section 3.2.1, a hybrid deployment of using ICN islands in an IP network is considered to be a realistic possibility for the uptake of ICN. We therefore use an IP backhaul in our baseline scenario. A native ICN backhaul will be covered in our sensitivity study.

In resource-contained IoT applications the transmitted data is typically small. For the payload size we also want to obtain the 'correct' order of magnitude. In the 'smart home' case, sensor data be

transmitted in a single message requiring typically a few bytes. Only the raw measurement data is transmitted, since the data itself can be identified by the URI/name. We therefore use an application layer payload size of 5 bytes.

Node characteristics
Based on the memory size of a polular Zigbee module [58], we set the cache size for the IoT node to 32 kB, which corresponds to roughly 600 NDN data packets. For the IP caching subscenarios, we use the same cache sizes on the gateway routers. For the sensitivity study we will also consider cases where the cache sizes can only store a low number of data messages, to see how this affects the network performance.

‘Smart home’: Covered differences between ICN and IP

Most of the identified differences between ICN and IP from Section 1.4.2 may be addressed by this scenario. The advantage of having fixed-length IP addresses compared to possibly longer length ICN names, may be verified by examining the used network bandwidth. The advantage of having a stateful forwarding plane in ICN may in this scenario be evaluated, since the Zipf popularity distribution is used. This increases the chance that multiple users are interested in the same data, allowing stateful forwarding techniques such as interest aggregation to come into action. The third difference we identified, relates to application layer vs. network layer caching. In this scenario, the advantage of having network layer caching may be demonstrated in subscenario S6 with an ICN backhaul network. Network layer caching allows here to deploy caching on nodes in the backhaul network. In this ‘smart home’ scenario, we will cover topologies where ICN-enabled IoT islands are used in combination with an IP backhaul. This allows us to evaluate the drawback in terms of bandwidth usage of having to send ICN messages over an IP network.

Table 3.2: *Smart home* baseline and sensitivity study parameter settings.

	Baseline						**Sensitivity study**					
Used L3-technology in IoT islands	**IP**		**ICN**				**IP**		**ICN**			
Subscenario name	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
Caching	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓
Backhaul network	IP	IP	IP	ICN	IP	ICN	IP	IP	IP	ICN	IP	ICN
Number of IoT islands	32						-					
Number of sensors per IoT island	25						-					
Number of IoT nodes of type (CL1, CL2, CL3)	(2,2,0)						(2,0,0), (0,2,0), (0,0,2), (5,0,0), (10,0,0), (15,0,0), (20,0,0)					
Data freshness period (s)	N/A	900	N/A	N/A	900	900	N/A	60, 120, 300, 600, 1200	N/A	N/A	60, 120, 300, 600, 1200	60, 120, 300, 600, 1200
Content popularity distribution (Zipf(α))	0.64						0, 0.5, 0.86, 1, 5					
Request frequency (requests/h)	12						-					
Data payload size (bytes)	5						-					
Cache size (packets)	0	600	0	0	600	600	0	0, 5, 10, 25, 50	0	0	0, 5, 10, 25, 50	0, 5, 10, 25, 50

‘Smart factory’: baseline scenario and sensitivity analysis

We use this scenario to compare the network performance of ICN and IP for a special use case where IoT nodes are also interested in data from other nodes. This corresponds to the operations optimisation use case described in Section 3.1. The data transmission frequency is typically higher in ‘smart factory’ scenarios and there are more strict requirements on data freshness. The role of the IoT islands is different for this use case, since we use the fact that IoT deployments in industrial environments use multiple islands to cover different areas in the same factory. This difference will mainly be reflected in the calculation of the content popularity distribution. As for the ‘smart home’ scenario, we will cover both caching subscenarios as well as subscenarios where caching is not used. For this specific scenario, the subscenarios without caching represent delay-sensitive optimisation processes, which require fresh and therefore non-cached data. The subscenarios with caching represent delay tolerant optimisation processes, which do allow the use of cached data. The used parameters are indicated in Table 3.3.

Topology

In a ‘smart factory’ scenario very high numbers of sensors are deployed on a small area. We reflect this in our choice for the number of IoT nodes per Islands, by using the highest possible nodes per island. The chosen evaluation platform described in Section 4.1.2, allowed us to use 50 IoT nodes per island in this scenario. The number of islands was also set to the highest possible number, which equals five, when 50 IoT nodes are deployed per island. We reduced the size of the backhaul network to 20 nodes to resemble a small corporate network.

Traffic patterns

In this ‘smart factory’ scenario, we consider the case whose goal is to optimise certain production processes. IoT nodes are in this scenario continuously generating and exchanging information with other nodes in the production process. For the consumer locations parameter we therefore focus on consumers which are located inside the IoT Islands (CL3). We set the number of consumers to 25 for our baseline scenario to allow reasonable experiment durations for the baseline and most sensitivity studies. With considerable effort we also completed simulations for 30, 40 and 50 consumers in the sensitivity analysis.

We set the data freshness duration to one second, because sensor data in these production environments is typically volatile. The content popularity distribution is now calculated over all available contents in the IoT Islands, since the IoT nodes now can request data from multiple other sensor islands. We set the request transmission frequency to one request per second since industrial applications typically use high update frequencies [59]. The backhaul network no longer represents the internet, rather it now represents a factory backhaul network. A realistic random graph will be used to represent this smaller corporate network. For the payload we again use a size of 5 bytes.

Node characteristics

The cache size is again set to the RAM size of the earlier mentioned ZigBee module, which is 32 kB [58]. This corresponds to roughly 600 NDN data packets.

‘Smart factory’: Covered differences between ICN and IP

The ‘smart factory’ scenario the one that may benefit more of an ICN approach rather than an IP approach. Compared to the ‘smart home’, this scenario may also address the benefit of having caching on the IoT sensor nodes themselves. The network layer caching enables in this case that an Interest may be satisfied by caches at four distinct locations. The Interest may be satisfied by the IoT node’s own cache, the gateway router of the IoT island of the consumer, one or multiple caches in the backhaul network or a cache at the gateway router of the producer.

Table 3.3: ‘Smart factory’ parameter settings.

	Baseline scenario						**Sensitivity analysis**					
Used L3-technology in IoT islands.	**IP**		**ICN**				**IP**		**ICN**			
Subscenario name	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
Caching	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓
Backhaul network	IP		IP	ICN	IP	ICN	IP		IP	ICN	IP	ICN
Number of IoT islands	5						1, 2, 10, 20, 50					
Number of sensors per IoT island	50						250, 125, 25, 12, 5					
Number of IoT nodes of type: (CL1, CL2, CL3)	(0, 0, 25)						(0,0,10), (0,0,20), (0,0,30), (0,0,40), (0,0,50)					
Data Freshness period (s)	N/A	1	N/A	N/A	1		N/A	0, 5, 30, 60, 300	N/A	N/A	0, 5, 30, 60, 300	0, 5, 30, 60, 300
Content popularity distribution (Zipf(α))	0.64						0, 0.5, 0.84, 1					
Request frequency (requests/h)	3600						-					
Data payload size (bytes)	5						-					
Cache size (packets)	0	600	0	0	600		0	0, 10, 100, 1000, 2000	0	0	0, 10, 100, 1000, 2000	0, 10, 100, 1000, 2000

3.2.4. Key Performance Indicators (KPIs)

According to the creators of the ICN paradigm, the use of named data and caching should allow for a more efficient use of network resources, while also providing a higher content delivery performance [3]. Since the IoT is expected to connect billions of devices to the internet, a resource-efficient network paradigm is needed to cope with the corresponding traffic increase.

To test which of the two network paradigms (ICN or IP) is most efficient for IoT, we need to define metrics or key performance indicators (KPIs) that are relevant for the IoT. In this study we will use five KPIs.

- End-to-end delay
- Hop count
- Network usage
- Cache hit ratio
- Cache capacity usage

A relevant aspect to measure when comparing ICN and IP is content delivery performance. ICN may be able to reduce content delivery times by caching data close to consumers, since the requested data may be cached close by, resulting in lower content delivery times. We verify this claim by measuring the *end-to-end delay* which we define as the elapsed time between the transmission of the data request and the reception of the corresponding data. Additionally, we also measure the *hop count*.

Since the IoT is expected to lead to the production and distribution of enormous amounts of data, our next metric will be the *average network usage*. The average network usage is calculated as a sum of all bytes transmitted on each link divided by the total simulation time. When, for example, a 1 kB packet travels over five links, then the corresponding average network usage can be calculated to be 5 kB divided by the simulation time. We compute the average network usage for both ICN and IP in all scenarios to verify to what extent ICN may help to reduce the used network resources.

Furthermore, we will use two metrics related to caching to examine the difference between IP caching (centralized, at the application layer) and ICN caching (distributed, at the network layer). The *cache hit ratio* metric is the ratio between the number of requests satisfied by a producer and the number of requests which are satisfied by a cache. The *cache capacity usage* metric, represents the time-averaged usage of available cache storage. For example, a cache capacity usage of 50% may indicate that the cache was on average half full during the experiment.

4

Implementation

The goal of this chapter is to describe the work done to implement the comparison scenarios from Chapter 3. Specifically, in section 4.1, we identify multiple evaluation platforms and we select the most suitable platform for our comparison. Section 4.2 describes the process of implementing the comparison scenarios in the chosen evaluation platform.

4.1. Evaluation platform selection

In order to compare ICN and IP under the scenarios defined in Chapter 3 and measure the KPIs described in Section 3.2.4, we need to evaluate those scenarios in a realistic platform which enables experiments of a reasonable scale. We will first compare multiple evaluation platforms in Section 4.1.1, and use this overview to find the best evaluation platform for our comparison in Section 4.1.2.

4.1.1. Overview of available evaluation platforms

We identify two different types of platforms: testbeds and simulators/emulators, which we will describe individually.

Testbeds

A testbed is a small-scale deployment of, in this case, IoT sensor nodes which allow to gather real-life measurement results, by using real applications and networking stacks. The biggest advantage is the representativeness of the measurement results, since, compared to simulations, typically fewer assumptions and simplifications are done to perform experiments. The biggest drawbacks are the typically fixed topology and hardware implementations of the testbed. Moreover, working with testbeds also places limits on the flexibility of the scenarios which can be evaluated, as well as on the size of the network that can be considered. The costs of setting up and maintaining a testbed can also be an important drawback. We only consider open and actively maintained IoT testbeds. In [60] and [61] an overview is given of multiple open IoT testbeds. Of the testbeds mentioned in those studies, only two testbeds are still operational at the time this assessment was done.

FIT IoT Lab

The FIT IoT-LAB testbed [62] is a large scale open testbed containing 2331 nodes at eight different locations in France and Germany. The facilities can be used free of charge. The nodes are positioned at fixed locations and have varying performance. Three types of sensor nodes can be used at different locations, each having different computational capabilities. All nodes use the IEEE 802.15.4 standard as their physical layer protocol and are connected in a mesh topology. Users can upload their custom firmware directly to the sensor nodes via a web page and schedule experiments when resources allow.

w-iLab.t

The w-iLab.t testbed [63] comprises 60 IoT nodes deployed in a mesh topology. The testbed is located in Zwijnaarde, Belgium and allows researchers to have full control over the testbed facilities. Similar

to the FIT IoT Lab testbed, users are able to upload their own custom firmware to the web-portal. The IoT nodes support the use of IEEE 802.15.4 and Bluetooth standards. Additionally, 16 IoT nodes are connected to mobile robots which can move freely between the stationary nodes.

Simulators and emulators

Typical advantages of simulators over testbeds are flexibility and extensibility. New features can be quickly added at relatively low costs and the possibility of running parallel simulations speeds up experiments execution considerably. The biggest drawbacks are the potential loss of precision and relevance caused by (over)simplifications and modelling. Another challenge is introducing randomness into simulations, needed when performing statistical analyses. As done for the testbeds, we only consider simulators and emulators which are still actively maintained by their creators or community.

OMNett++

The OMNett++ discrete-event network simulator [64] is written in C++ and uses separate functionality modules for simulations. The most important functionality module is the INET framework which implements common network stacks and physical layer protocols. It is one of the few simulators with a graphical user interface. One of the externally available modules called CCN-lite [65] provides an ICN implementation for the simulator. The authors indicate that their implementation is far from complete and does require extra work for real use.

Cooja: The Contiki Network Simulator

Cooja [66] is a simulator which emulates IoT nodes running the Contiki operating system. It therefore allows to run simulations with real-life network stacks and software implementations. The Cooja network emulator supports simulation of wireless sensor networks connected in a mesh topology and is not built to also simulate larger interconnected sensor domains. The main drawback of this simulator, in relation to our study, is that only an outdated and experimental CCNx implementation is available for the Contiki operating system [66], which is used by the Cooja simulator. Moreover, there is hardly any documentation available describing the functionality and use of this software.

miniNDN

The miniNDN emulation tool [67] is an extension to the well-known Mininet software [68]. MiniNDN uses virtualisation to emulate multiple NDN hosts on a single machine. This has implications on the size of the networks being tested since all nodes are virtualised by a single machine. The emulator uses the complete NDN software suite built and maintained by the NDN consortium. Wireless networks are not modelled well by miniNDN, since virtual ethernet links are used to model connections.

ndnSIM (NS-3)

The developers of the ndnSIM (NS-3) simulator [69, 70] managed to integrate the main building blocks of the NDN implementation: the *Named Data Networking Forwarding Daemon* (NFD) and the *Named Data Netowrking c++ library* (ndnx-cxx), into the popular NS-3 simulator, while still preserving nearly all functionalities. This adaptation creates the possibility to use NDN as a L3-protocol with the various networking protocols and technologies already integrated in NS-3. The discrete-event simulator provides support for heterogeneous network simulation, which also allows the simulation of larger backhaul networks. Users need to write simulation scripts in C++ to define the topology and network stacks used.

NS-3 DCE

NS-3 DCE (Direct Code Execution) [71] is a special version of NS-3 which is able to execute kernel space or user space networking protocols directly in the simulator. It is then possible to use the existing implementations for the NDN architecture in a simulation scenario. This direct code execution framework does provide extra flexibility at the cost of having less optimised simulations. This raises scalability issues and limits simulations to a small number of nodes.

Icarus
The Icarus simulator [72] is written in Python and is built to verify and test the performance of caching strategies. The simulator does not implement a specific ICN architecture and networks are modelled at a high level of abstraction. This also implies that the Icarus simulator currently is not capable of simulating real ICN deployments for IoT applications.

4.1.2. Evaluation platform selection

We select the most suitable evaluation platform for our study based on the support for the defined comparison scenarios in Chapter 3. The evaluation platform should be able to model the chosen networking technologies for the defined topologies. Moreover, it should be possible to evaluate the comparison scenarios with the key performance indicators from Section 3.2.4. We analysed for each platform whether these requirements are met (Table 4.1).

Looking at Table 4.1, it becomes clear that the Icarus emulator and the MiniNDN software are not suitable for our comparison due to the lack of support for our selected physical layer protocols. If we also consider the necessary IP technologies, we can conclude that only the FIT IoT LAB testbed, the w-iLab.t testbed, the ndnSIM (NS-3) simulator and the Cooja emulator allow the use of the three selected networking technologies.

From the mentioned platforms, an ICN implementation is only available for both testbeds and the ndnSIM (NS-3) simulator. The simulator uses the complete software package built and maintained by the NDN consortium, while the testbeds use the CCN-Lite [65] implementation.

Furthermore, the evaluation platform should support experiments with IoT islands connected via a realistic backhaul network, as we defined in Chapter 3. In an IoT island, wireless IoT nodes connect to a central gateway router. This gateway router connects to other IoT domains via a wired network. Table 4.1 indicates that IoT island topologies can be evaluated with the FIT IoT LAB testbed. Topologies with wired backhaul networks used in our comparison scenarios are however not supported. The NS-3 based ndnSIM (NS-3) simulator does provide the tools to model IoT island topologies and backhaul networks. So the ndnSIM (NS-3) simulator is the most suitable platform for our comparison so far.

The last row in Table 4.1, lists the evaluation platform support for the KPIs defined in Section 3.2.4. It can be seen that all platforms are able to perform measurements with our used metrics, including the ndnSIM (NS-3) simulator. Based on the arguments given above, we conclude that the ndnSIM (NS-3) simulator is indeed the most suitable platform for our comparison study, since it meets all our requirements. We will therefore use this platform for our study.

Table 4.1: Evaluation platform features comparison.

		Testbed		Simulator			Emulator		
		FIT IoT LAB	w-iLab.t	OMNet++	ndnSIM (NS-3)	NS-3 DCE	Icarus	Cooja	MiniNDN
L1 technology	IEEE 802.15.4	✓	✓	✓	✓	✓	✗	✓	✗
IP features	6LoWPAN	✓	✓	✓	✓	✗	✗	✓	✗
	IPv6	✓	✓	✓	✓	✓	✗	✓	✓
	CoAP	✓	✓	✗	✓	✓	✗	✓	✓
ICN features	NDN/CCN stack	CCN-Lite	CCN-Lite	(CCN-Lite)	NDN	CCNx	✗	✗	NDN
Topology support	IoT islands	✓	✗	✓	✓	✓	✗	✗	✗
	Backhaul network	✗	✗	✓	✓	✓	✗	✗	✗
KPI's	Network usage	✓	✓	✓	✓	✓	✓	✓	✓
	Hop count	✓	✓	✓	✓	✓	✓	✓	✓
	End-to-end delay	✓	✓	✓	✓	✓	✓	✓	✓
	Cache hits	✓	✓	✓	✓	✓	✓	✓	✓
	Cache utilisation	✓	✓	✓	✓	✓	✓	✓	✓

4.2. Simulation scenario implementation

This section will describe the steps taken to implement the comparison scenarios from Chapter 3 in the NS-3 based ndnSIM simulator which was chosen in Section 4.1. NS-3 uses separate modules to model individual network technologies and applications. A simulation script can be used to combine these

submodules into one custom simulation model, which can be used to perform measurements. The goal of this section is to describe the creation of these simulation scripts and to present the necessary configurations and adaptations to the simulator.

We will first describe, in Section 4.2.1 and Section 4.2.2, the work done to implement the IP stack and NDN stack specified in Section 3.2.2 in the simulator. In Section 4.2.3, we wil describe how we modelled hybrid ICN deployments where ICN islands are connected via IP networks. The methodology of setting up simulations will be presented in Section 4.2.4.

4.2.1. IP stack implementation

We will describe the IP stack implementation using a bottom-up approach where we start at the physical layer, and move towards to the application layer. We follow the comparison scenario design described in Section 3.2.1 and Section 3.2.2.

L1: Physical Layer + L2: MAC layer

IEEE 802.15.4 physical layer

The IoT Islands need to use the IEEE 802.15.4 communication protocol at the physical layer. We implement this specific technology by using the lr-wpan NS-3 module [73]. This module follows the 802.15.4-2006 standard, which prescribes operation in the unlicensed 868-868.6 MHz (Europe), 902-928 MHz (USA) and the global 2.4 GHz frequency bands. The NS-3 model uses the 2.4 GHz band, allowing a maximum data rate of 250 kbit/s. The channel is modelled as an additive white Gaussian noise (AWGN) channel and the modulation method used follows the standard, which defines the use of offset quadrature phase-shift keying (OQPSK).

Furthermore, the standard defines two different classes of devices. The first type is called the full function device (FFD), which can be used in any topology and can act as the core of a star network. The second type is called the reduced function device (RFD), which uses a low-profile stack implementation and can only be used as a leaf node. Every network can have one personal area network (PAN) coordinator, which must be a FFD. This device typically serves as the center of the network providing coordination services to network nodes. The lr-wpan NS-3 model does not (yet) model different device roles and uses the FFD-role for all devices.

In our simulation model, we use one single gateway per IoT island with both a 802.15.4 radio and a wired network interface to connect to other networks. The IoT sensor nodes are placed in a grid pattern and are all in range of the central gateway. Nodes can communicate directly with other nodes in the same IoT island without the need to contact the gateway first.

IEEE 802.15.4 MAC layer

At the MAC layer, there is currently only one single MAC implementation available for the lr-wpan module [73]. This MAC layer assumes the radio to always be in 'listening' mode when the node is not transmitting. In constrained IoT applications it is common to switch off the radio when no packets need to be received or transmitted, in order to save energy. It is important to use a smart strategy to decide when to go into sleep mode, since no packets can be received when a node is sleeping and packet loss may occur.

MAC layer protocols which allow sleeping patterns are not yet available for the NS-3 simulator. There is currently one radio duty cycling (RDC) MAC protocol in development, which is based on the ContikiMAC RDC protocol [74]. This MAC protocol in development also comprises the implementation of an energy measurement toolbox for the 802.15.4 protocol, which supports the simulation of energy consumption at the radio level. The module is currently being reviewed by the NS-3 maintainers and will likely be integrated in a future NS-3 release. We merged the latest version of this unfinished module with the ndnSIM simulator, to see whether we already could use it for our study. We managed to successfully deploy this implementation in the ndnSIM simulator and we solved two bugs in the model's finite state machine implementation. However, after several tests with large topologies we came to the conclusion that this module is not yet suitable for our study, since it affected the simulator's stability and performance. Therefore we decided to leave energy usage experiments as future work since it may provide intersting results related to power requirements of the more advanced stateful forwarding plane and complicated message format of NDN. For this study we use the default MAC implementation [73].

Backhaul network

We model the backhaul network by using an external topology generator called BRITE [75]. According to its creators, BRITE was developed to provide a universal topology generator capable of generating representative internet topologies for multiple simulation platforms.

We use BRITE to generate the random graphs for the backhaul networks of our covered scenarios. In the 'smart home' scenario, the backhaul network should represent an internet like topology and in the 'smart factory' scenario it should represent a corporate network.

To model the internet-like backhaul networks we use BRITE's top-down model [75]. This model uses two hierachical topologies, which comprise an autonomous system (AS) topology and a router-level topology, shown in Figure 4.1. An autonomous system can be seen as a subpart of the internet under a single administrative control, which can be for example an internet service provider [76]. First the top-level AS topology is generated, where each node represents a single AS network. Then for each AS node, a different random graph model is used to generate the autonomous system inside network. The IoT islands are connected to a randomly selected leaf node in one of the router level topologies.

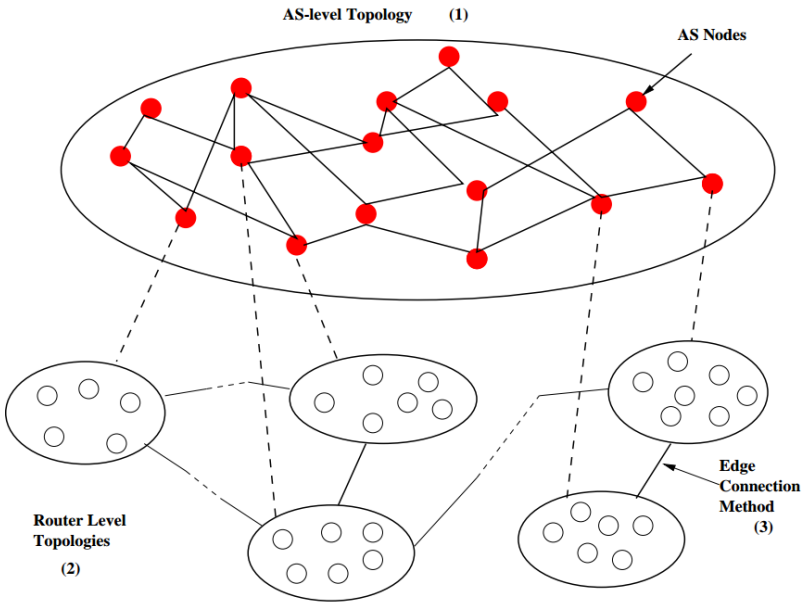


Figure 4.1: The used top-down internet topology. Picture from BRITE manual [75].

To model the topologies, we can choose from two different types of random graphs, the Barabási–Albert graph [77] and the Waxman graph [78]. We select the Waxman graph to model the router level topologies, since it is considered to be a representative model for intra-domain networks [79]. The 'smart factory' scenario only uses this router level topology. The Waxmann graph can be constructed by placing N nodes randomly on a two dimensional plane. The probability of having a link between nodes u and v can then be calculated by Expression (4.1):

$$p(u,v) = \beta e^{\frac{-d}{\alpha L}} \tag{4.1}$$

where

α = Waxman parameter $(0 \leq \alpha \leq 1)$
β = Waxman parameter $(0 \leq \beta \leq 1)$
d = Euclidean distance between nodes nodes u and v
L = Maximum euclidean distance between any two nodes in the graph.

Expression 4.1 indicates that, given a set of node locations, the Waxman graph requires two parameters α and β. The value of α determines the relative number of short links and long links. A lower value for α results in many relatively short links. The parameter β correlates to the link density of the graph. When

β is increased, then the link density of the graph increases. The real value of α and β is dependent on the modelled autonomous system. In [80] maximum likelihood estimation is used to to estimate these parameters for a number of real-life networks from the Internet Topology Zoo database [81]. We use the parameter estimation of the UUNet network, to calculate suitable values for the BRITE configuration. We calculated that the corresponding Waxman parameters for our simulation set-up are $\alpha = 0.64$ and $\beta = 0.47$.

For the modelling of the AS-level topology, which we additionally need for the 'smart home' scenario, we use the Barabási–Albert graph. The model uses an incremental approach to add nodes one by one to a graph. In this graph type, new nodes are more likely to connect to nodes with a higher degree[1]. A newly added node u is connected to m existing nodes with a probability given by Expression 4.2.

$$p(v) = \frac{d_v}{\sum_j d_j} \tag{4.2}$$

where

d_v = Degree of node v.
$\sum_j d_j$ = Sum of the degree of all nodes in the currently existing network.

For our simulations we set $m = 2$, to model the internet's AS topology [82]. So every new node will initially connect to two nodes, with a specific probability defined by Expression 4.2. When the AS level graphs and the intra-domain graphs are generated, the next step is to connect both graphs. A link in the AS-level topology does not map directly to a specific node-to-node connection on a router level. The BRITE topology generator combines both topologies by connecting two randomly selected nodes from two AS domains which are connected in the AS-level graph. The used BRITE configuration file can be found in appendix B.1.

L3: Networking and Routing

6LoWPAN

We use the 6LoWPAN module from the NS-3 simulator [83], to act as an adaptation layer between the 802.15.4 MAC layer and the IPv6 networking layer. We used the default module settings, which include HC1 header compression [45] and UDP checksum compression.

IPv6

On top of the 6LoWPAN layer we use the NS-3 IPv6 module [84]. We used the default module settings, with the exception of some parameters of the Neighbour Discovery protocol (NDP). The NDP is an advanced version of Address Resolution Protocol (ARP) in IPv4, and its task is to discover the roles and L2 addresses of neighbouring hosts and routers. Hosts send neighbour solicitations messages to discover L2 addresses of a host in the same subnetwork. The addressed host responds with a Neighbour discovery message to exchange L2 addresses. The L2 address is cached to reduce the transmission of Neighbour Solicitation (NS) messages. The entry is considered stale after a fixed time, and a new NS message must be transmitted. While experimenting with the NS-3 IPv6 module, we noticed that the neighbour advertisements are sometimes not correctly interpreted causing unnecessary packet loss. We prevented this from happening by increasing the REACHABLE_TIME parameter in the IPv6 implementation. This reduces the number of needed Neighbour Solicitation messages since the neighbour discovery entries are valid for a longer time.

Routing

As mentioned in Chapter 3, we do not focus on comparing the efficiency of routing of IP against ICN. Rather, in our study we make the assumption that routing information is known before the simulation of the scenario starts. This could be achieved when global routing is used in the NS-3 simulator. Unfortunately, global routing is not available for IPv6 simulations and is only available for IPv4. We therefore had to implement our own solution. Specifically, we use an initialisation time before a simulation run starts. During this time a real routing protocol is used to populate the routing tables. When the simulation starts, the routing protocol is switched off and the routing table entries are used for the rest of

[1] The degree of a node equals the number of edges incident to a node.

the simulation. This feature was implemented using the only IPv6 routing protocol currently available in NS-3 called RIPnG [85]. We updated the implementation to allow more frequent gratuitous Response Message transmissions. Moreover, we changed the RIPnG module to support larger networks by increasing the fixed maximum hop count.

L4-L7: CoAP applications

In Section 3.2.2, we decided to use the CoAP protocol as the application layer protocol for the IP scenarios. This protocol is currently not part of the NS-3 simulator and we therefore had to build our own implementation.

We based our design on the udp-echo-server [86] and udp-echo-client [87] model scripts, part of the NS-3 simulator. The UDP client script must be adapted to allow the transmission of CoAP GET messages which should follow the Zipf popularity distribution specified in Section 3.2.3. The CoAP server application should be able to receive CoAP requests and to respond with the appropriate CoAP response message. We also had to build a special CoAP gateway application, which is able to cache CoAP messages at the IoT Island gateway router.

CoAP client

The CoAP client needs to send a CoAP GET message for the requested content, for example, GET/TEMP to the correct CoAP server. Our modelling approach is to calculate the total size of a CoAP GET message with a known payload and to add this to the payload of the UDP packet. We have calculated the size of this CoAP GET message by following the CoAP message format specified in the CoAP IETF standard [51], which is illustrated in Figure 4.2. The size of the non-optional header can be calculated by adding the sizes of the fields from the first row from Figure 4.2. This first row contains the CoAP protocol version (2 bits), the message type (2 bits), the token length (4 bits), the request code which indicates the message type (8 bits) and the message id (16 bits) which is used to identify duplicate messages. So the total size of the compulsory header equals 4 bytes.

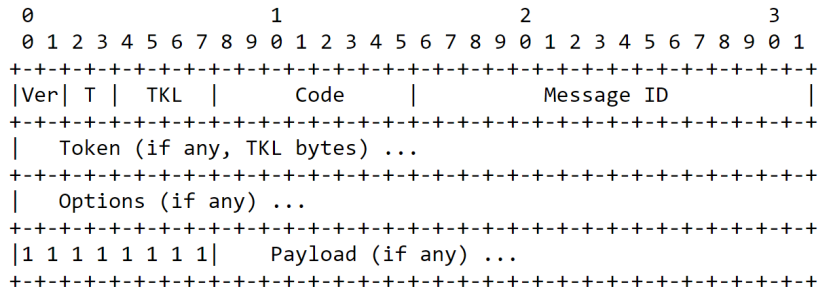


Figure 4.2: CoAP Message format. Picture from IETF CoAP standard [51].

We also have added the optional token field to total header size, since the standard requires the CoAP GET messages to have a token which is used to match requests and response messages. The token should be unique for every concurrent transmission between a source/destination pair. The standard specifies a variable token size of 0-8 bytes which can be chosen based on the application specifics. Smaller tokens result in a lower protection against request-response mismatches, while a longer token causes extra overhead. The CoAP IETF standard [51] prescribes that a token should be at least 4 bytes long when the node is connected to the internet. This should improve security by reducing the risk of response spoofing [51]. We therefore choose to use a token length of 4 bytes, which is also sufficient to prevent duplicate tokens in our scenarios. The options field should contain the URI of the requested content. Which has a one byte identifier and a variable URI size. The last row containing the end-of-options indicator (0xFF) and payload field are not used in a CoAP request message. We can now conclude that a CoAP request message with a GET method is 8 bytes long, excluding the URI. For the URI we use ”SensorData/x” which we also use a a name for the ICN scenarios to keep the comparison fair. The x denotes the requested content number. The payload which we will add to the UDP message will be 8 bytes + size(URI) long.

The CoAP client requests content based on the chosen Zipf popularity distribution. The ndnSIM simulator provides example applications for NDN simulations which use a Zipf-Mandelbrot popularity

distribution [88]. We integrate the functions related to the generation of a Zipf distribution into the CoAP client application. The last part of the URI, the requested content number, is defined by the Zipf distribution.

CoAP server
The CoAP server application needs to receive CoAP request messages and then needs to send a CoAP response packet with the requested data. We calculate the size of this packet by summing the compulsory header of 4 bytes, token of 4 bytes and the end-of-options market of 1 bytes. This results in an overhead size of 9 bytes. So the UDP payload for a CoAP response should be 9 bytes + size(requested content) long.

CoAP caching gateway
The CoAP IETF standard [51] also provides a description on the use of caching CoAP response messages at dedicated CoAP nodes. A CoAP caching gateway may satisfy a CoAP request if it has a cached CoAP response message available. The CoAP standard also prescribes the use of a freshness technique similar to ICN's data freshness duration. A CoAP server may specify for each CoAP response a data freshness duration after which the response should be considered stale.

We implemented the CoAP caching gateway, by modifying our CoAP server application. The new application will cache every incoming CoAP response message until the cache reaches its full occupancy. The Least Recently Used (LRU) cached entry will then be replaced if a new CoAP response comes in. We selected these strategies, because the NDN implementation uses the same caching policies. The CoAP caching gateway application is installed on all IoT island gateway routers, when we consider cases with IP caching enabled.

4.2.2. NDN stack implementation

We will also describe the NDN stack implementation using a bottom-up approach where we start at the physical layer, moving towards the application layer. We again follow the comparison scenario design described in Section 3.2.1 and Section 3.2.2.

L1: Physical Layer + L2: MAC layer

We use exactly the same topologies and backhaul network as we did for the IP case described in Section 4.2.1.

L3: Networking and Routing

The NDN layer can directly be deployed on top of the 802.2.15.4 MAC layer and does not require any special adaptations. As we mentioned in Section 4.2.1, we do not want to compare the efficiency of the routing protocols available for NDN and IP. Therefore, we use the global routing option available for the NDN stack. The FIBs from all nodes are populated with the shortest routes to the content producers, before a simulation is initiated.

L4-L7: ICN applications

The NDN consumers use the 'Zipf-Mandelbrot' application [88] to transmit Zipf distributed interests. Very few changes were needed to deploy the Zipf-Mandelbrot application for our scenarios. We used the names which are similar to the URI's we used for the IP scenarios. Every producer can supply a single data packet with name "SensorData/x" where x again will be replaced by a draw from the Zipf distributed content list. The NDN producers use the default producer application [89].

4.2.3. ICN over IP

In our scenarios we cover two different types of ICN deployments, a native ICN case where both the backhaul nodes and the IoT Islands are using ICN and a hybrid scenario where ICN islands are connected via an IP backhaul. The latter case is considered to be more realistic in the short term, since it does not require the entire internet to support ICN protocols at the network layer. In this section we will first introduce several ICN over IP deployment configurations, subsequently we will describe our ICN over IP modelling approach.

ICN over IP deployment configurations

A common solution to allow the transmission of ICN packets over the internet, is the use of UDP tunneling [90]. The interconnection of ICN islands is then accomplished by manually setting up UDP tunnels. ICN interests or data messages are encapsulated in UDP packets, traverse the internet and are unpacked at the receiving ICN gateway node. Figure 4.3 illustrates this encapsulation process. The NDN testbed [6] is currently using this type of UDP tunneling. A more advanced solution is proposed in [41], where software-defined networking is used to automatically configure UDP tunnels between ICN domains.

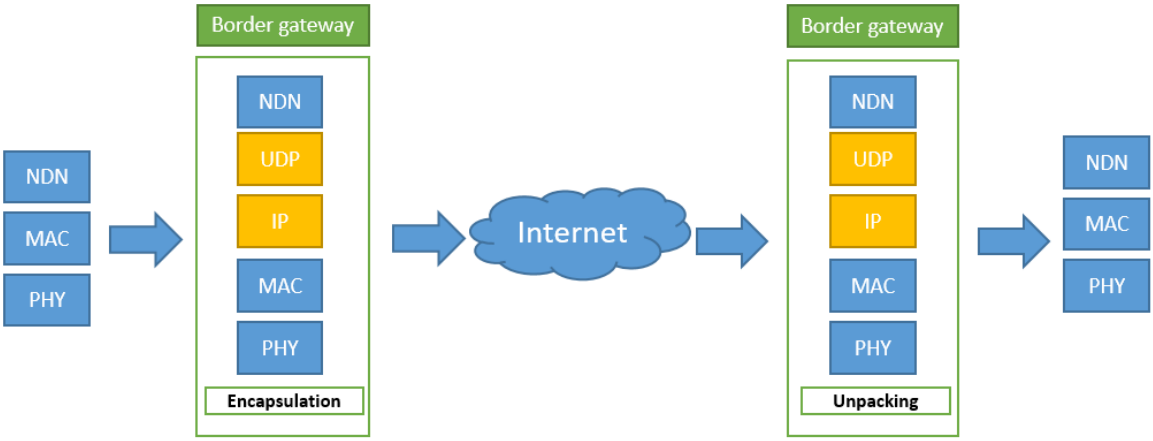


Figure 4.3: UDP encapsulation of NDN packets between two L2 ICN islands.

A different approach was recently presented by Cisco called Hybrid ICN (hICN) [91], where ICN packets are structured in such a way that they comply with the IP packet format. All ICN features such as name-based forwarding and reactive caching are preserved and the IP address is used as a name. IP routers can forward hICN messages in exactly the same way as normal IP packets, since there is no difference in the packet format. Only special routers equipped with a hICN forwarding module are able to perform ICN specific features such as caching and interest aggregation. This work is still in an early phase and implementation details, as well as experimental results have not been disclosed.

In our simulation study, we will focus on UDP tunnelling. Since only a high-level design of hICN is released, it is not possible to simulate and compare this approach in detail and we leave this as future work.

Modelling approach

Initially, the opportunity of modelling the real encapsulation process of NDN messages into UDP packets was investigated. It was found that UDP tunnelling is currently not supported by the NS-3 simulator and a new module had to be built. The UDP encapsulation process of NDN messages therefore needs to be implemented from scratch. Moreover, the dynamic translation between names and IP addresses in simulation scenarios would also require the design and implementation of a DNS-like application.

We used a different modelling approach with a similar accuracy and representativeness, but with a more efficient implementation process. The native NDN deployment case where all nodes are NDN compatible, was taken as a baseline for the encapsulation modelling. The baseline deployment was modified in a way to best resemble a hybrid IP configuration. The modelling method enabled us to simulate UDP encapsulation without the need to build a name-to-IP translation module.

When an interest or data message encapsulated in a UDP packet is transmitted over an IP network, it will not be cached and no forwarding decisions will be based on the ICN messages enclosed. We therefore disable caching for all backhaul nodes for this configuration. The second important feature of ICN which will not be available in ICN-as-an-overlay networks is the aggregation of interest messages. When multiple interests arrive at the same node for the same content, a single PIT entry will be generated for both interests. When a matching data packet arrives, the node will forward this packet to both interested consumers. As mentioned earlier, this aggregation is not available when a data or

interest message is enclosed in an UDP packet and we need to disable this feature for the backhaul nodes. We implemented this by adding an extra random name component to the interest message when the node is forwarded by a gateway node. Two interests for the same data are then unlikely to be combined in backhaul PIT nodes, since they will have a different extra name component with high probability. he extra added name component is removed when the ICN message arrives at the gateway node connecting the backhaul to the ICN islands.

The encapsulation in UDP packets also results in extra packet overhead due to the required extra headers. We therefore have to add this extra payload to the interest and data messages. For this purpose, we reuse the extra name component to resemble the extra data overhead. We set the size of the name component to be equal to the size of a IPv6 header and a UDP header combined. We define this name component size to be 48 bytes, since an IPv6 header is at least 40 bytes and a UDP header is 8 bytes.

4.2.4. Simulator set up

Simulation duration settings

For both the NDN and IP simulations the data producers and consumers applications are started after 120 seconds. We do this to allow our custom routing solution the necessary time to fill all routing tables for the IP cases. We verified that the routing protocol indeed converges after 120 seconds for all covered scenarios. At t=120s, the applications start transmitting and receiving packets.

The simulation durations is fixed to 50,000 seconds of simulated time. We analysed the convergence for the covered metrics and we found that steady state behaviour can be observed for all our scenarios after 10,000 seconds of simulation time. We therefore only use measurement data from 10,000 to 50,000 seconds in our parsing scripts [92].

Confidence intervals and random number generator

We run all simulations scenarios ten times with different random seeds and use the obtained performance results to derive 95 percent confidence intervals. A fair comparison between ICN and IP requires that all simulation aspects are the same for ICN and IP for a given scenario. This also results in the fact that all simulation aspects which are randomly chosen such as the topologies, should be the same for ICN and IP for equal random seeds. So when we run ten iterations of one IP scenario, the same simulation environment should be used when we run ten iterations of the ICN scenario. We accomplish this by using the substream feature of the NS-3 simulator. When declaring a random variable in the simulator, it is possible to assign a specific substream to this specific variable. When two instances of the same simulation with the same random seeds are used, a fixed random substream will ensure that the same random numbers will be sampled for both cases. We use this feature for all randomly chosen scenario aspects such as topology and transmission frequency, to ensure equal configurations for both ICN and IP.

4.2.5. Source code

To ensure reproducibility of the results, we released our used simulation scripts to the public [93]. The simulation script which can be seen as the top level entity combining the different used and developed modules, is composed out of three main files. These files can be found in the ”Scratch/wsn-iot-v1”-folder. The *wsn-iot-v1.cc* file contains all configurations for both the IP as well as the ICN scenarios. Functions applicable to both ICN and IP are implemented in the ”g_header.cc”-file. All ICN related stacks, functions and configurations are included in the *ndn-header.cc* file. The separate IP related implementations can be found in *ip-header.cc*.

5

Results and analysis

The goal of this section is to present the results of the simulation study and to analyse the observed trends. As discussed in Section 3, we have defined two main scenarios covering a ‘smart home’ use case and a ‘smart factory’ use case.

5.1. Smart home

For the ‘smart home’ scenario, specified in Section 3.2.3, we simulated both a baseline scenario and performed a sensitivity analysis. We first describe the baseline scenario results in Section 5.1.1 and discuss the sensitivity analysis results in Section 5.1.2.

5.1.1. Baseline scenario

We start by describing the results for the baseline ‘smart home’ scenario.

End-to-end delay

We present the end-to-end delay results in a so-called violin plot which can be seen in Figure 5.1. The violin plot uses kernel-density estimation (KDE) to illustrate the underlying distribution of the plotted data. Compared to the empirical probability density function, the KDE uses kernels to smooth the resulting estimate [94]. The KDE is rotated and mirrored on two sides of a vertical axis. Our violin plots also show the 90th percentile, 10th percentile and average value of the data for every covered scenario. We will first compare the covered subscenarios where caching is disabled and we will cover the subscenarios with caching enabled in a separate paragraph.

No caching

The IP and ICN violins show two distinct regions in the KDE. A lower part which is centred around 8 ms and a stretched region with a peak around 40 ms. This ‘smart home’ baseline scenario assumes consumers located in the backhaul network (CL1) and consumers located at the gateway routers (CL2). The delays in the lower region around 8 ms originate from consumers close to or at the gateway, and the stretched higher region comprises delays experienced by nodes in the backhaul network.

If we compare the IP with the ICN subscenarios, we see that the use of an ICN network instead of an IP network does not significantly reduce the average, 10th delay percentile and 90th delay percentile. There is also no clearly noticeable difference between the ICN cases with either an IP or ICN backhaul.

Caching enabled

If caching is enabled for the IP case, we see a reduction of 3 ms for the average delay. This reduction can be explained with help of the KDE. If we compare the IP with caching violin with the IP without caching violin, we notice that the KDE is stretched to 0 ms if caching is enabled. Since the IP caches are only deployed on the gateway routers, this 0 ms delay can only occur when a gateway router issues a request for data which is available in its own cache. This can be the case if data has earlier been requested by other consumers. The KDE peak at the delay region around 8 ms can be attributed to

requests originating from the gateway router which cannot be satisfied from the cache. Furthermore, we see that the upper delay region, residing above the average, has slightly more weight at lower delay values compared to the IP without caching subscenario. Requests from consumers in the backhaul network may be satisfied from the gateway routers cache, eliminating the need to forward the request to the IoT nodes over the constrained 802.15.4 interface.

When caching is enabled for the ICN cases we see similar effects occurring for the ICN with IP backhaul subscenarios, as for the IP subscenario. The KDE is stretched to 0 ms and weight of the upper delay region is moved to a lower value. We also see a comparable reduction for the average delay of 3 ms when caching is enabled for the ICN subscenarios. The ICN subscenario with ICN backhaul shows the largest average delay reduction of 5 ms when caching is used, compared to the ICN subscenario without caching. In this specific subscenario, backhaul nodes are also able to interpret and cache ICN messages and no encapsulation overhead is needed. An Interest message may therefore also be satisfied by a backhaul node.

If we compare the IP with caching subscenario with the ICN with caching subscenarios, we can conclude that an ICN deployment with IP backhaul does not improve the delay significantly. When the backhaul is implemented using ICN, the deployment of ICN then may allow a minor average delay improvement of just 3 ms compared to IP. Moreover, the 90th delay percentile is in this case reduced by a modest 1 ms if ICN with caching is deployed. In this ‘smart home’ scenario such a small end-to-end delay reduction may not be relevant due to the scenario specifics, which do not require low delay values.

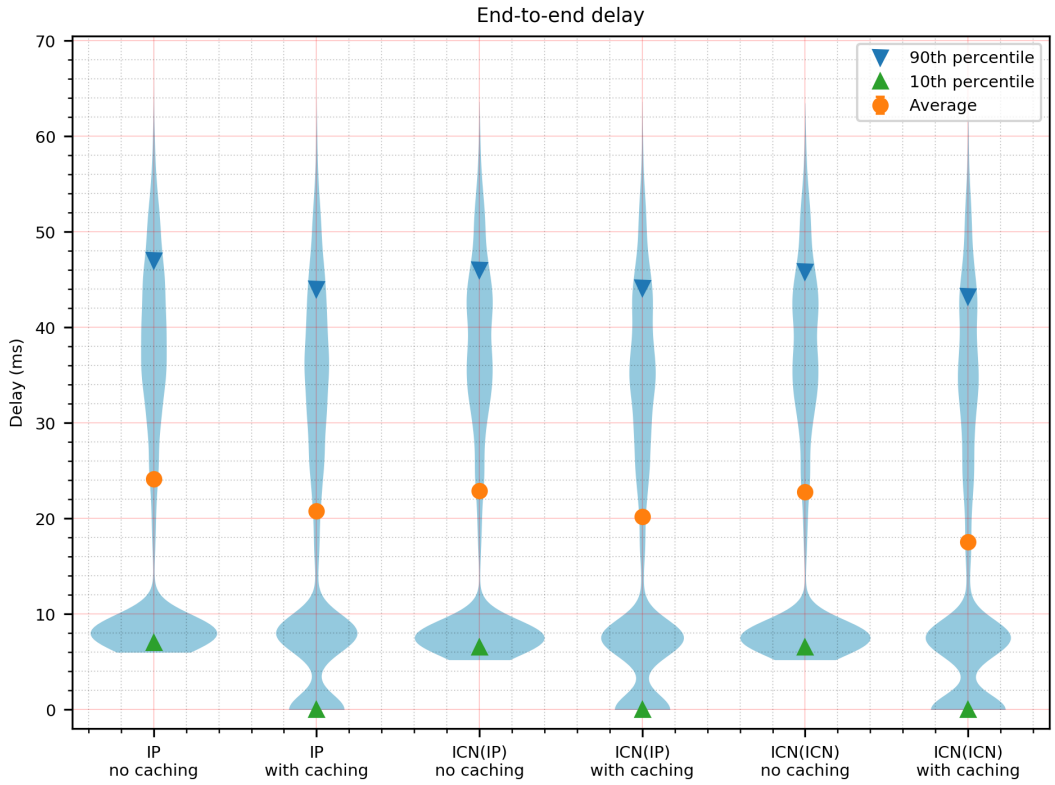


Figure 5.1: Delay results for the ‘smart home’ baseline scenario. The blue shape shows the kernel density estimate (KDE) for the delay distribution. The error bars indicating the 95% confidence intervals for the average delay are not visible due to their small size.

Hop count

We also considered the hop count metric for this ‘smart home’ scenario. The results for this graph are very similar to the delay results since they are strongly correlated. The hop count can therefore be seen in the appendix in Figure C.1.

Network usage

We will now consider the network usage for this 'smart home' scenario, from which the results can be seen in Figure 5.2. We defined the average network usage in Section 3.2.4 as the sum of the total number of bytes transmitted on each link divided by the simulation time.

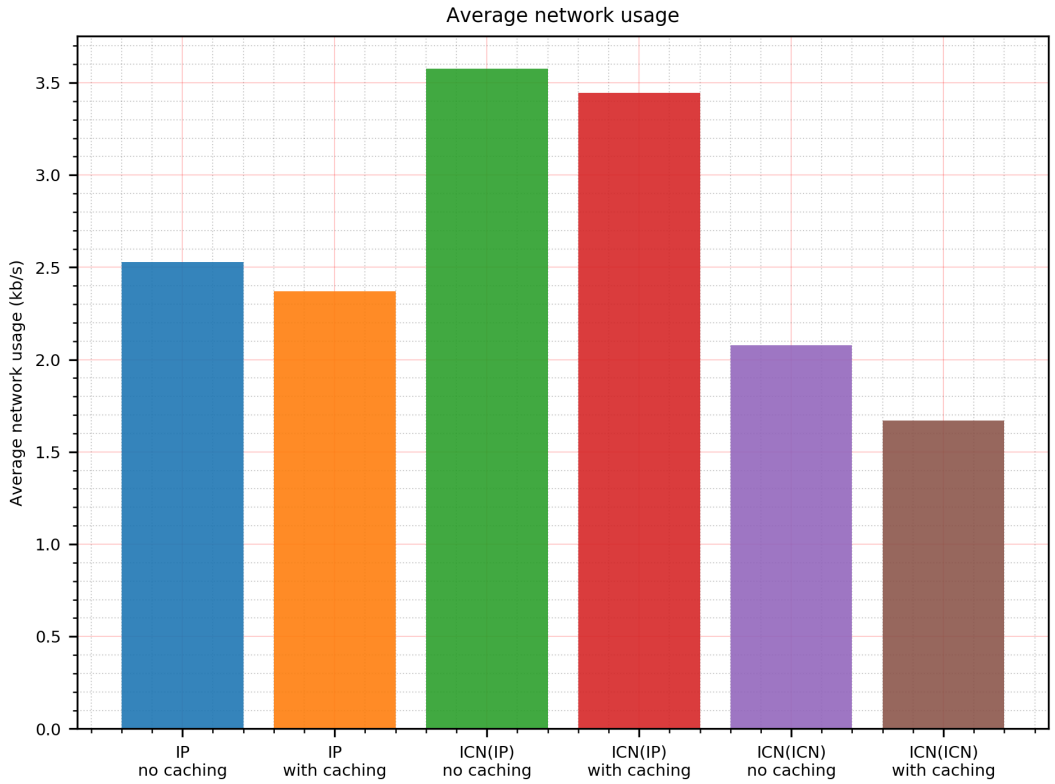


Figure 5.2: Network usage results for the 'smart home' baseline scenario.

No caching

For the subscenarios with disabled caching, it can be seen that the network usage is significantly higher for the ICN(IP) network compared to the native IP deployment. The average network usage is for this case increased by 44% if an ICN(IP) implementation is used instead of an IP deployment. This increase is mainly caused by the overhead introduced by the UDP encapsulation in the backhaul network. When we compare the IP subscenario without caching with the ICN(ICN) deployment without caching, we do see a significant reduction of 32% in network usage when the ICN(ICN) network is used.

Caching enabled

We see only a small reduction of 6% in the average network usage when caching is enabled for IP compared to the native IP case. This is a reasonable result since the IP caching deployment only uses caching at the gateway routers. This caching technique may reduce the number of hops by a maximum of one, when the transmission between the gateway router and the IoT nodes is not needed due to a cache hit.

For the ICN subscenarios with IP backhaul, we also see a network usage reduction when caching is enabled compared to the ICN(IP) network without caching. The reduction for this case is 4%. A larger reduction in network usage can be seen for ICN(ICN) subscenarios. The use of caching for these subscenarios results in an average network usage reduction of 20%. Since backhaul nodes are able to interpret and cache ICN messages, a cache hit in the backhaul may save multiple transmissions to the IoT islands thereby reducing the average network usage.

As discussed in Section 1.5, the average network usage is a very important metric for IoT applications, since it is expected that a very large number of IoT devices will be connected to the internet.

From the results in Figure 5.2 we can conclude that for this ‘smart home’ scenario, an ICN deployment with ICN backhaul is the most resource efficient solution. The ICN(IP) subscenario is considered to be the most realistic short-term ICN deployment which can be easily connected to the current internet. For the average network usage metric, we can say that such a deployment actually increases the average network usage.

Cache hit ratio

Six pie charts in Figure 5.3 show the results for the cache hit ratio, which we defined in Section 3.2.4 to be the ratio between the number of requests satisfied by a producer and the number of requests which are satisfied by a cache.

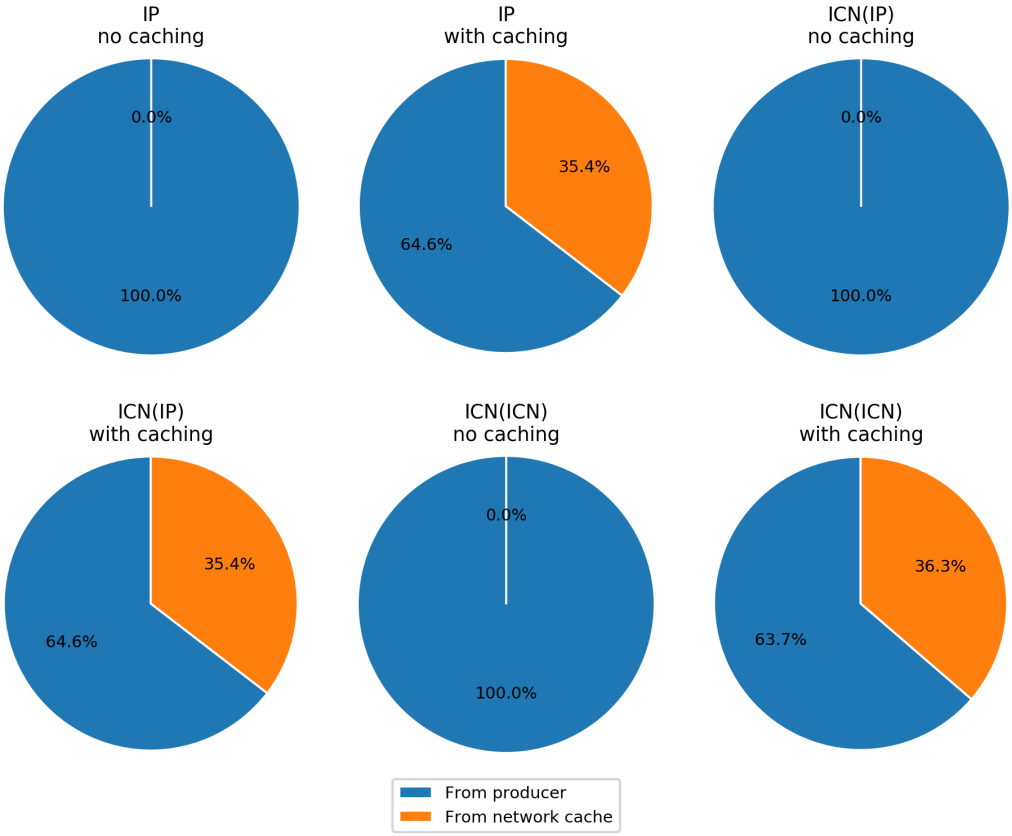


Figure 5.3: Cache hit ratios for the ‘smart home’ baseline scenario. The width of the confidence intervals for these results is +/- 0.2%.

No caching

The pie charts presenting the cache hit ratio for subscenarios where caching is disabled show a trivial result: all requests/Interests are satisfied by a producer.

Caching enabled

For the subscenarios with caching enabled, it can be noticed that the cache hit ratio is comparable at about 35% for all cases with caching. For the IP and ICN(IP) subscenarios this is a logical result, since the consumers in this ‘smart home’ scenario either reside at the gateway router or in the backhaul network. For both IP and ICN(IP) a request/Interest can then be satisfied by the gateway router if it has the data available or the producer in the IoT island. The request/Interest will not be satisfied in the backhaul since these subscenarios do not use caching in the backhaul network. Due to this similarity and the equal caching strategies, a similar cache hit ratio can be observed for these subscenarios in this ‘smart home’ scenario.

When we do have caching in the backhaul with the ICN(ICN) subscenario we see that the cache hit ratio only improves by about 1% compared to the ICN(IP) subscenario. So it can be said that the probability that a request/Interest can be satisfied by a cache is nearly equal for all subscenarios, but that the performance improvement is mainly influenced by the location of the caches which satisfy the requests/Interests.

Cache capacity usage

In Section 3.2.4 we defined the cache capacity usage as the time-averaged usage of available cache storage. For the IP subscenario with caching enabled a single bar is used to indicate the cache capacity usage at the IoT island gateway routers. For the ICN subscenarios, the average cache capacity usage is given for nodes in the backhaul network, gateway routers and IoT nodes. For the caches in the backhaul we calculate the average over all caches which stored one or more data packets during the simulation. We also plotted the maximum cache capacity usage, which is the highest observed cache capacity usage during the simulation. The results for this 'smart home' scenario can be seen in Figure 5.4.

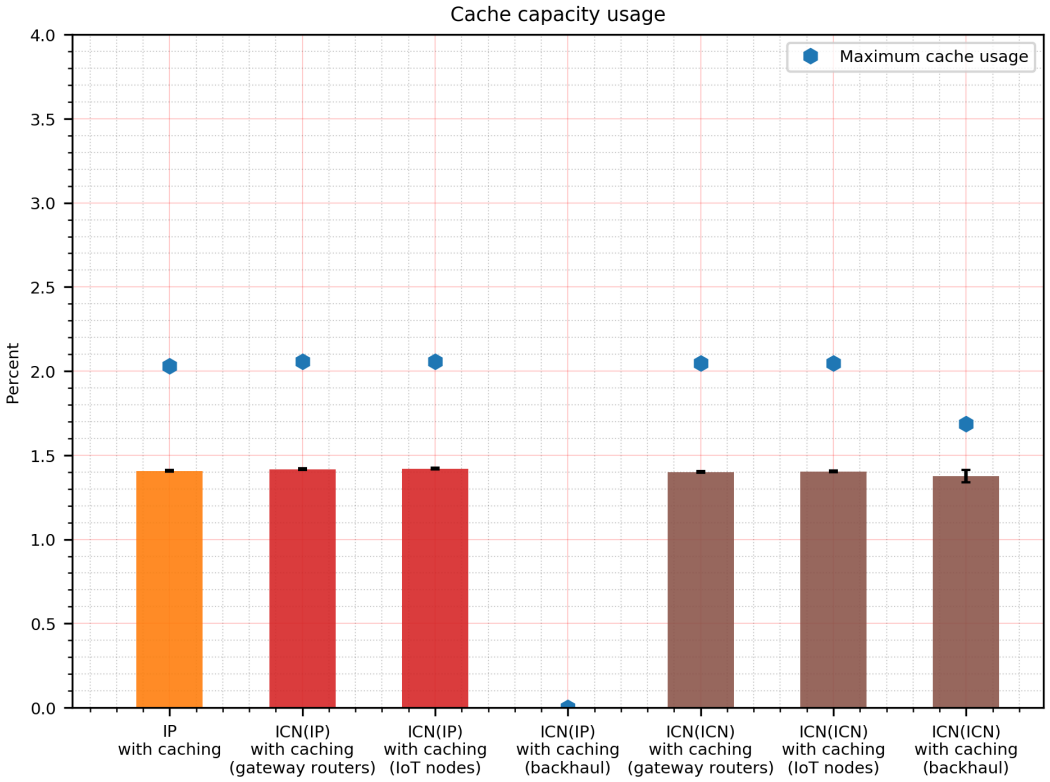


Figure 5.4: Cache capacity usage results for the 'Smart home' baseline scenario. The black error bars indicate the 95% confidence intervals for the average cache capacity usage

No caching

When caching is disabled there will obviously be no cache capacity usage. For clarity we therefore left out these results in Figure 5.4.

Caching enabled

If we look at the subscenarios with caching, we see that only a very small percentage of the cache capacity is used. A cache size of 32 kB (600 packets) is therefore already more than enough for this 'smart home' scenario.

The IP subscenario with caching shows that only on average 1.4% of the cache capacity is used at the gateway routers. For the ICN(IP) case we a similar percentage for the average and maximum

cache capacity usage at the gateway. The selected ICN implementation, described in Section 3.2.2, uses a cache everything strategy to decide whether a data message should be cached. All incoming data messages received by the IoT nodes will therefore be added to the nodes' caches. This behaviour can be seen in the graph, since also the IoT nodes actively cache data. The backhaul nodes do not cache data in the ICN(IP) subscenario since the backhaul uses IP to forward packets.

The ICN(ICN) subscenarios show comparable results for the gateway routers and IoT nodes. The important difference is visible for the backhaul nodes which now also actively cache data. It must be noted that, as we already mentioned in Chapter 3, we also set the cache size of the backhaul nodes to size specified in the scenario. Backhaul nodes and routers are typically less constrained and have more storage available than our selected 32 kB (600 packets). However, the results show that very small cache sizes are already sufficient for this 'smart home' scenario.

5.1.2. Sensitivity analysis

In this sensitivity analysis, multiple parameters of the baseline scenario are varied, to test its effect on the KPIs in both the IP and ICN cases. In this section we will present the sensitivity study results from the 'Smart home' scenario. An overview of the covered scenario aspects can be seen in Table 3.2.

Consumer location

In Chapter 3, we have defined three consumer location types. Consumers with CL1 reside in the backhaul network, consumers with CL2 are situated on the gateway routers and consumers with CL3 are inside the IoT islands. Since the baseline 'smart home' scenario uses a combination of consumers at CL1 and CL2, we use this sensitivity study to see the effect of the consumer location on our covered metrics. The results for this sensitivity study are pictured in Figure 5.5.

CL1

When we look at the end-to-end delay for consumers at CL1 in the top left corner of Figure 5.5, we see that most of the subscenarios provide a similar delay with overlapping confidence intervals. The only exception is the ICN(ICN) deployment with caching enabled. This specific scenario provides the lowest delay. An explanation can be given with the help of the hop count graph. The average hop count is equal for all subscenarios where caching is disabled, since all requests can only be satisfied from the producers in the IoT islands. This difference in delay is not caused by a lower required number of hops, but only by technology-specific aspects such as packet sizes in the backhaul. For the IP caching and ICN(IP) caching subscenarios, we see a similar reduction in the average hop count, because both deployments can only benefit from caches deployed at the gateway routers. The ICN(ICN) subscenario with caching has the highest reduction in delay because also the lowest number of hops are required, due to the availability of caches in the backhaul. The average network usage in the top right corner of Figure 5.5 shows that the ICN(IP) deployment with and without caching has the highest network usage, as we discussed earlier, this is caused by the encapsulation of ICN messages into UDP packets. The IP subscenarios have a lower network usage. The ICN(ICN) subscenarios have the lowest network usage. From the cache hit ratio graph in the lower left corner, we see that consumers at CL1 have an overlapping ratio of satisfied requests/Interests. The average and maximum cache capacity usage graphs, show that the ICN(ICN) deployment has the highest cache capacity usage. This is caused by the backhaul nodes which may cache data for multiple IoT islands.

CL2

Gateway consumers experience a much lower delay compared to consumers at CL1, since the data can be requested from the producer within a single hop. The relative delay reduction when caching is used is much larger, since a cache hit prevents the need to send a request/Interest into the network. This is also confirmed by the fact that hop count is less then 1 for the caching subscenarios. The network usage is now in-line with the ordering of the delay and hop count results, because the requests/Interests and data messages do not have to traverse the backhaul network. The cache hit ratio graph and cache capacity usage graphs show similar results as for CL1.

CL3

Consumers with CL3 are located inside the IoT domain and will in this 'smart home' scenario request

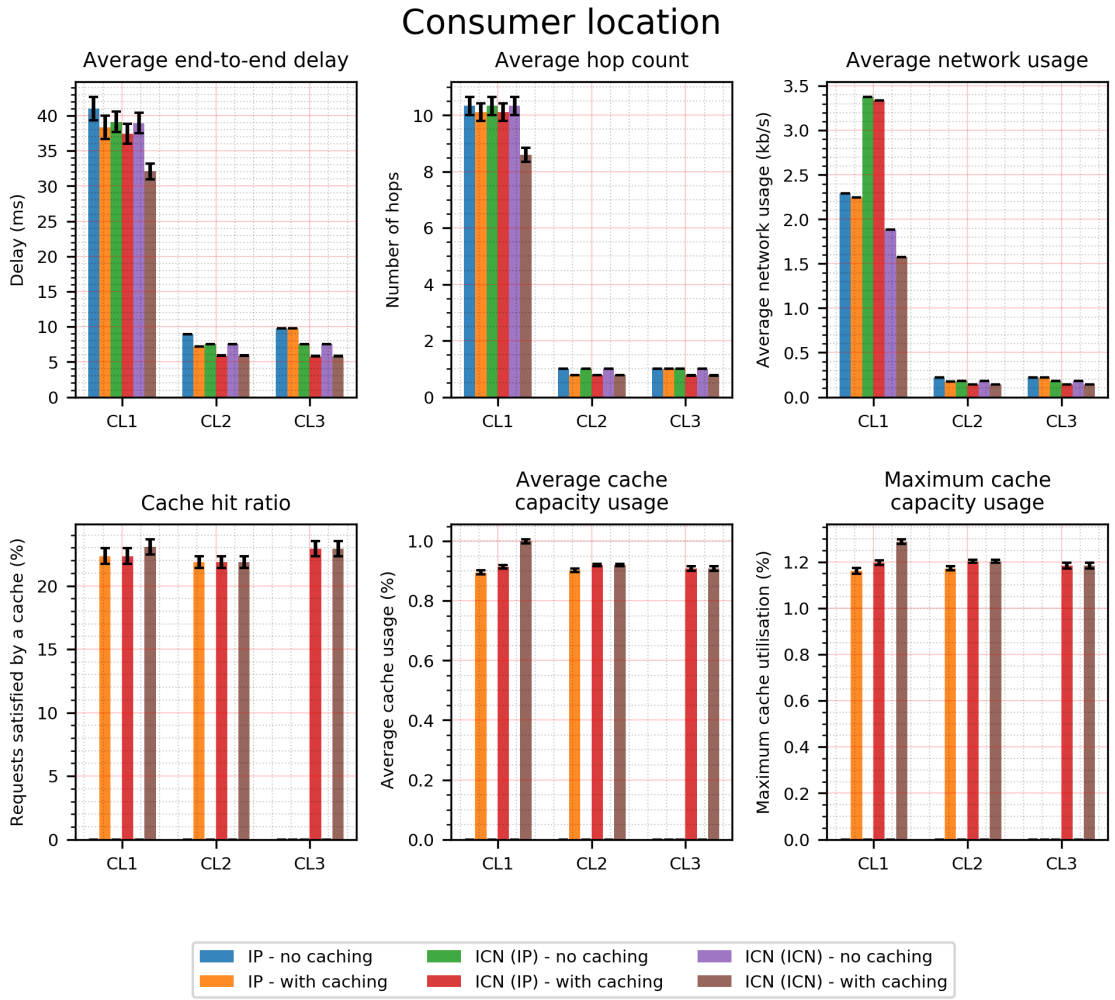


Figure 5.5: Sensitivity study results, with different data consumers varied. The 95% confidence intervals are indicated with error bars. Baseline scenario consumer locations: two consumers at CL1 and two consumers at CL2.

data from other IoT nodes. If we look at the end-to-end delay, it can be seen that the delays for both IP subscenarios are equal. IoT nodes can now directly request data from other IoT nodes without the need to first contact the gateway router. IP caching will therefore not provide a performance increase for consumers at CL3. Since requests/Interests originating from consumers inside the IoT domain do not have to cross the backhaul, it can be noted that the ICN(IP) results are equal to the ICN(ICN) results. If we look at the ICN subscenarios without caching we see that the end-to-end delay is reduced compared to IP, while the hop count is equal. When caching is enabled in the ICN subscenarios, a small reduction in delay and hop count can be observed, this is due to the fact that IoT nodes are able to cache data that they are relaying for other nodes, thereby increasing data availability within the IoT island. A future request originating from the application running on this IoT node, may then be satisfied from the data in cache, eliminating the need to forward the request into IoT island. The average network usage is again low because all traffic stays inside the IoT islands. ICN caching at the node level helps reducing the average network usage, due to the possibility of satisfying interests/requests from the node's own cache. From the cache hit graph we see that the cache hit ratio is again similar to the ratios observed for CL2 and CL1. Also for the average and maximum cache capacity usage we see similar results as for CL2.

Number of consumers

We also varied the number of consumers which are interested in the data of a 'smart home'. We limit this sensitivity study to consumers at CL1, since this is a realistic location for an increasing number

of consumers. The results are shown in Figure 5.6. When we look at the end-to-end delay, we see

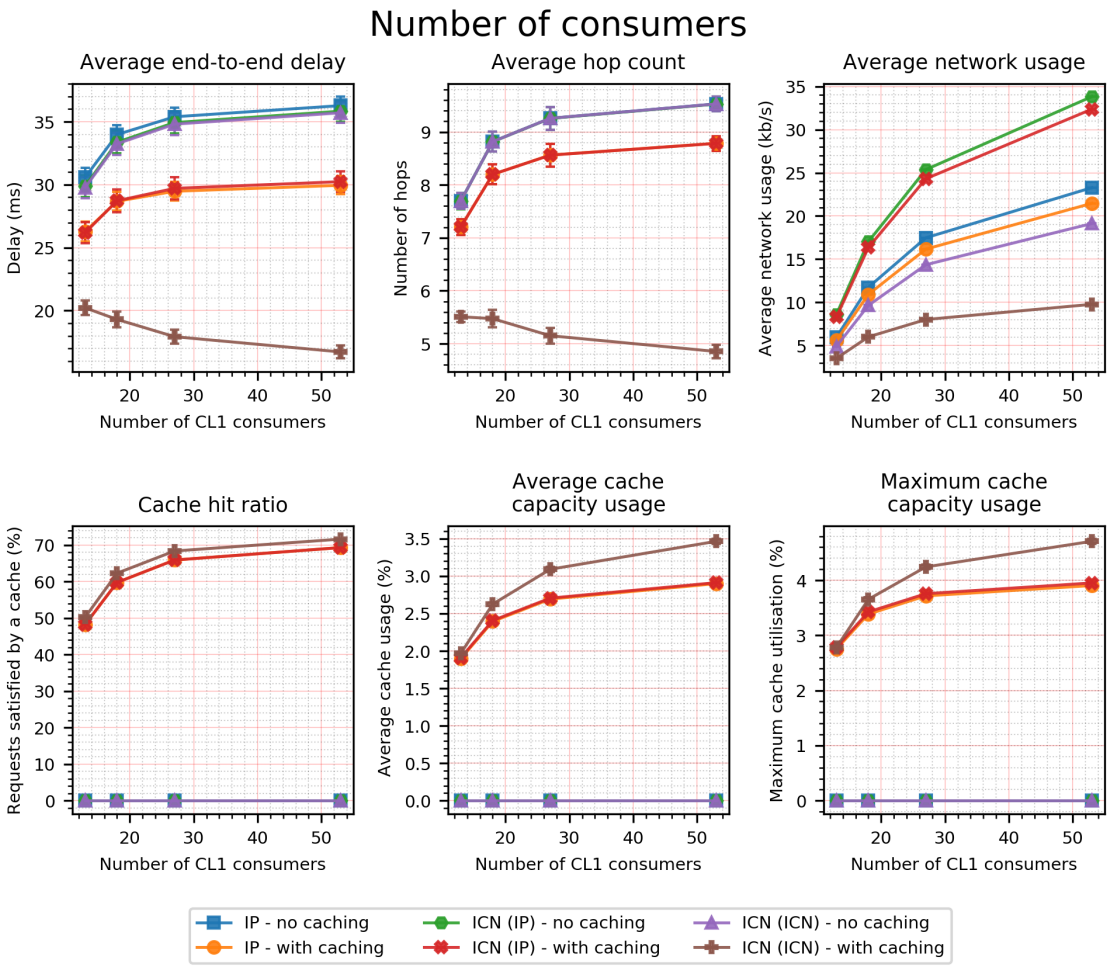


Figure 5.6: Sensitivity study results, when the number of CL1 consumers is increased. The 95% confidence intervals are indicated with error bars. Baseline scenario consumer locations: two consumers at CL1 and two consumers at CL2.

that the delay increases when the number of consumers increase for the IP and ICN(IP) subscenarios. An increase in the total number consumers will also cause an increase in the traffic load in the IoT islands, causing a busier channel and hence increased end-to-end delays. Deploying caching at the gateway routers does reduce the experienced latency. The ICN(ICN) deployment with caching behaves differently. From the graph it can be seen that the delay decreases when the total number of consumers increases. When more Interests are issued for the same data, the chance that an interest may be satisfied by a cache in the backhaul increases. This is also confirmed by the hop count which shows a decrease for increasing traffic load. The data may thus be found closer to the consumer. The average network usage graph shows a similar ordering of the subscenarios. When the number of consumers is increased we see that the ICN(ICN) deployment makes the most efficient use of the resources and therefore the network usages increases at a lower pace than for the other subscenarios. The average cache capacity usage shows that ICN(ICN) has on average the most data in cache. Backhaul nodes equipped with caches may store data from multiple ‘smart homes’ resulting in a higher cache capacity usage.

Data freshness period

Data may be cached for a duration specified by the data freshness period. After this time, the cached data is considered stale and the data is removed from the cache. We varied the data freshness period to see the effects on our covered key performance indicators. The results can be seen in Figure 5.7.

We start by looking at the end-to-end delay results in the top left corner of Figure 5.7. We first notice

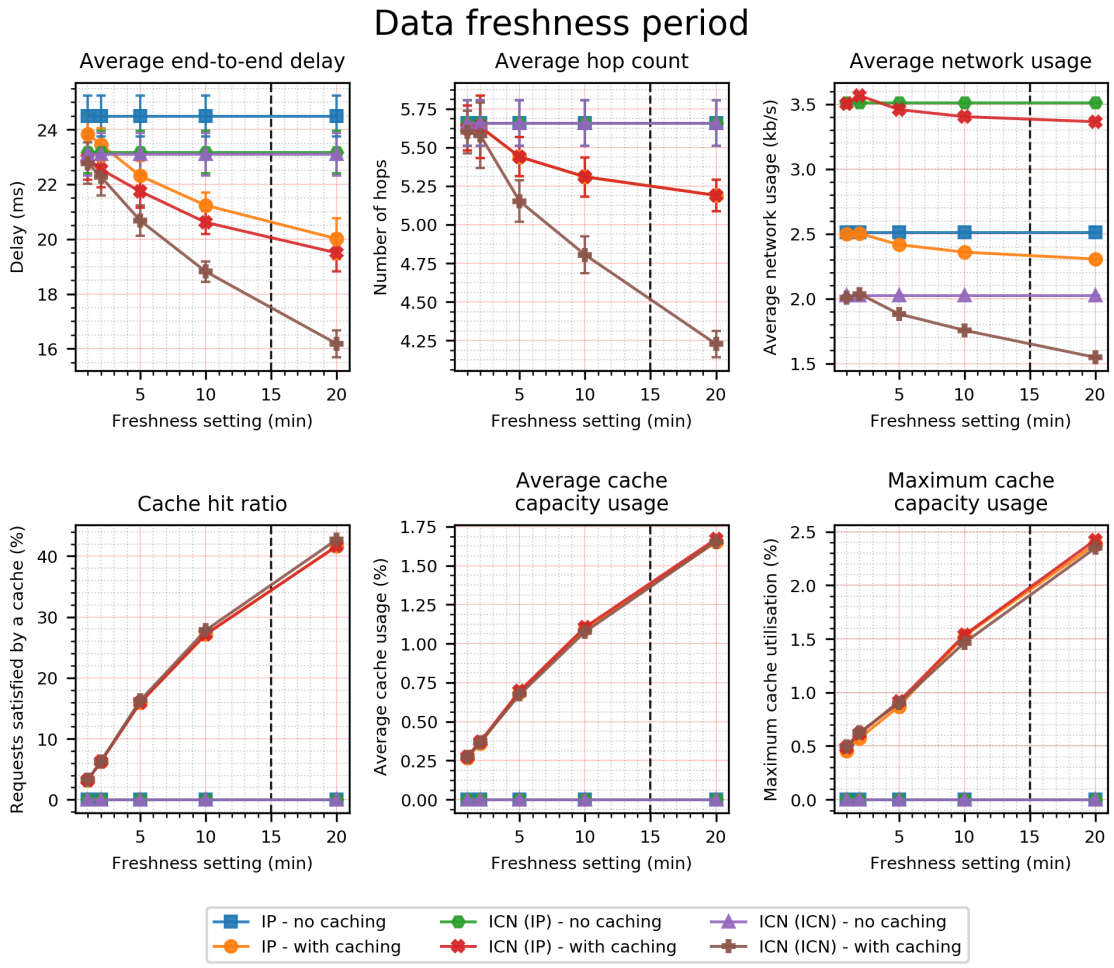


Figure 5.7: Sensitivity study results, when the data freshness duration is changed. The 95% confidence intervals are indicated with error bars. Baseline data freshness period is indicated with vertical dotted line.

the trivial result that the delay for the subscenarios where no caches are deployed, is not influenced by the data freshness period. If we look at the IP subscenario with caching, we see that the delay for a freshness setting of five minutes is already reducing the end-to-end delay by 2 ms. When the data freshness period is further increased and thus allowing data with a longer lifetime to satisfy the request, we see that the end-to-end delay is further decreased. The ICN(IP) subscenario has a similar delay course to the IP caching case, since the scenario specifics of having consumers in the backhaul and on the gateway combined with the use of a ICN(IP) usage, do not allow the specific foreseen advantages of ICN to come into action. When the freshness duration setting is increased in the ICN(ICN) subscenario, we see a steeper decrease in the end-to-end delay, since now also the backhaul nodes are equipped with caches.

Similar behaviour can be seen in the hop count graph. The IP with caching and ICN(IP) with caching networks need a similar amount of hops to satisfy the consumers requests/Interests. For the ICN(ICN) subscenario with caching we see that the hop count decreases at a higher rate due to the larger number of distributed caches in the backhaul.

The average network usage in the top right corner, illustrates that the use of an ICN(IP) network with or without caching uses the most network bandwidth for all covered settings of the freshness period. A lower network usage may be observed for the IP subscenarios and the ICN(ICN) deployments have the lowest network usage.

The cache hit ratio in the lower left corner provides an insight that the fraction of requests/Interests satisfied by a cache is equal for all freshness settings. We already saw in Section 5.1.1 that this also was the case for the baseline scenario. From the hop count graph it was clear that the ICN(ICN) has the

lowest required number of hops to satisfy the Interests compared to all other covered subscenarios. Nevertheless, we see in this cache hit ratio graph that the same number of requests/Interests were satisfied from a cache.

When we now look at the average and maximum cache capacity usage graphs, we immediately notice that the cache capacity is sufficient for all our covered freshness duration settings. When the data freshness duration is increased, we see that the average cache capacity usage is also increased. This can be explained by noting that a higher data freshness duration will allow data to stay in the caches for a longer time period. The caches will therefore on average be filled with more data.

Content popularity distribution

We will now vary the Zipf-parameter (α) from the used popularity distribution function described in Section 4.2.1. When α is set to zero, all data is equally likely to be requested. Increasing α will result in steeper probability mass function, where few data messages have a relatively high probability to be requested and a lot of data have small request probabilities which is common in the current internet. We varied α between zero and one, since these are realistic values for multiple applications as described in Section 3.2.3. From the baseline scenario in Section 5.1.1, we know that the selected cache size of 600 packets can be considered very large for this 'smart home' scenario. We initially conducted this sensitivity study with the baseline cache size and found that there was no significant effect observable when varying the Zipf parameter for our selected (realistic) values. This is most likely caused by the fact that the covered values of α, do not result in significant changes in the behaviour of the forwarding planes. We will therefore use a small cache size in this sensitivity analysis to see the effects of the interaction between the content popularity distribution and the cache replacement strategy. We therefore set the cache size to ten packets for the sensitivity analysis of the content popularity distribution. The results can be found in Figure 5.8.

From the delay graph it can be seen that a change in Zipf-parameter only has a small effect on the delay for this 'smart home' scenario. For the IP and ICN(IP) we see that the delay is not significantly changed by changing the Zipf-parameter. The ICN(ICN) with caching subscenario shows a delay reduction of 2 ms when α is changed from zero to one. When the Zipf parameter is increased more interests will be issued for similar data, which increases the chance of a cache hit. Since the ICN(ICN) network allows to cache in the backhaul, this cache hit may occur closer to the consumer. This effect is also confirmed by the hop count graph, which shows a decrease for increasing α.

From the average cache capacity usage graph we see that a higher value for α results in a lower average cache capacity usage. When we have a larger value for α, more requests will be issued for the same data. There will therefore also be a smaller diversity of data in the caches, causing a lower cache capacity usage.

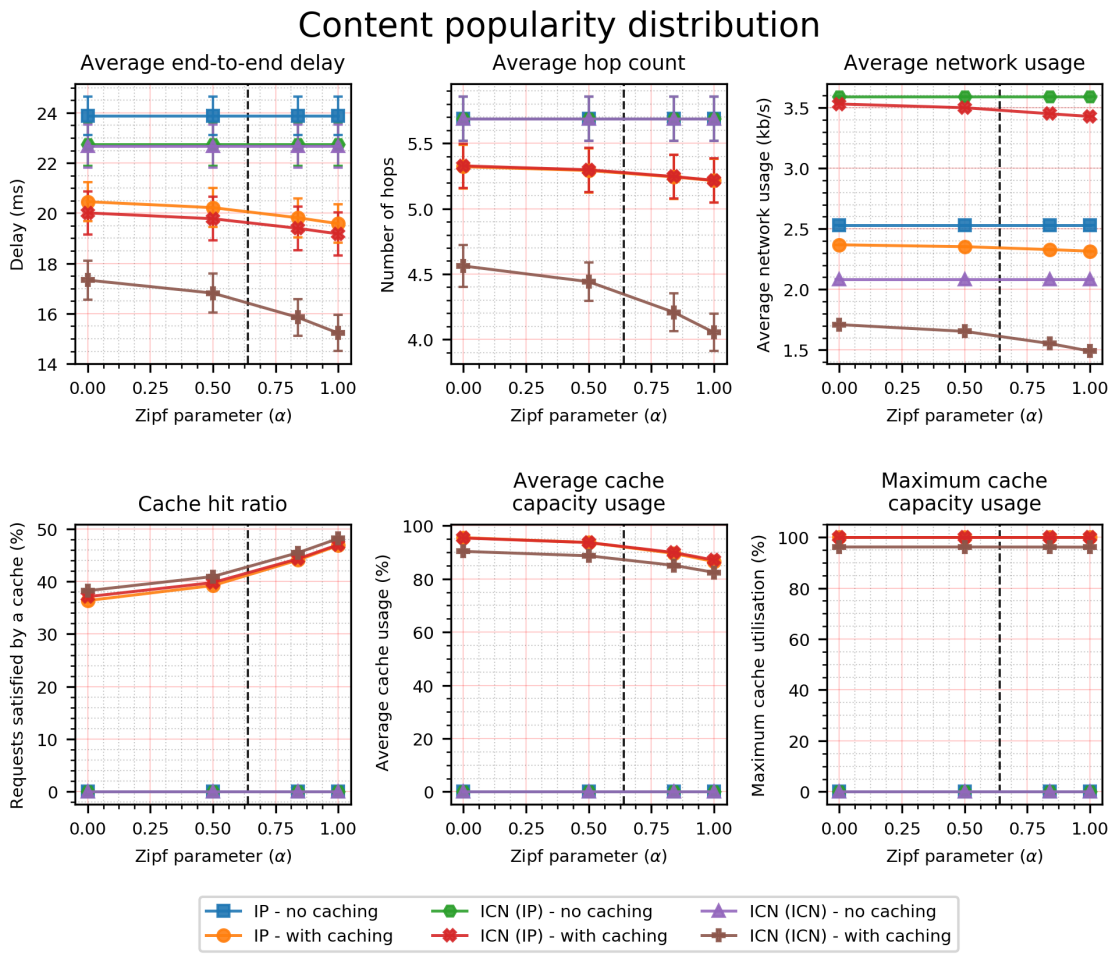


Figure 5.8: Sensitivity study results, when the content popularity distribution is changed. The 95% confidence intervals are indicated with error bars. Baseline value for Zipf parameter (α) is indicated with vertical dotted line.

Cache size

The last sensitivity study done for this 'smart home' scenario will cover multiple settings for the cache size. The results can be seen in Figure 5.9.

We use this sensitivity study to find the smallest cache size which is sufficient to provide maximum performance for both IP and ICN. From the cache hit ratio graph we see that a cache hit ratio is achieved of 42% when the cache size is set to ten packets, this ratio does not further improve when the cache is enlarged. If we look at the hop count and delay graph, we can draw a similar conclusion that a cache size of ten packets is already enough for this 'smart home' scenario. The minimum cache size which allows maximum caching performance in this 'smart home' scenario must be of a minimal size of ten packets. As discussed earlier, the average and maximum cache usage are presented as a percentage the maximum cache size. This explains the shape of the cache capacity usage graphs, which now show the relative cache capacity usage for increasing cache sizes. Increasing the cache size beyond the 'minimum' size of ten packets, will result in a lower cache capacity usage percentage since the same data messages will be stored in a larger cache.

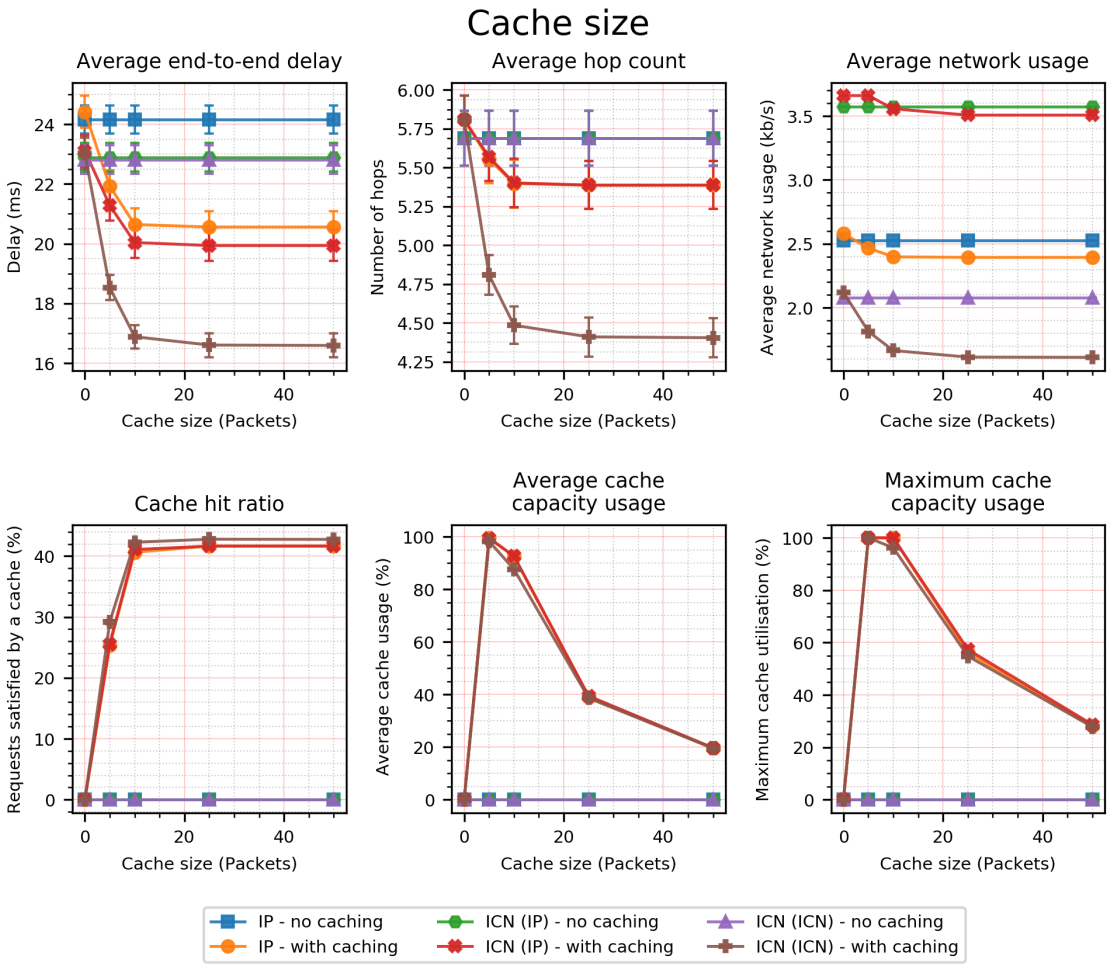


Figure 5.9: Sensitivity study results, when the cache size is changed. The 95% confidence intervals are indicated with error bars. Baseline scenario cache size 600 packets.

5.2. Smart factory

For the 'smart factory' scenario, specified in Section 3.2.3, we performed both a baseline scenario and a sensitivity analysis. We first describe the baseline scenario results in Section 5.2.1 and we will present the the sensitivity study results in Section 5.2.2.

5.2.1. Baseline scenario

Similar to the 'Smart home' scenario in Section 5.1, we start by discussing the results of the baseline scenario. We discuss the results for every covered KPI individually.

Delay

We present the end-to-end delay results in a so-called violin plot, showing the results of the six different covered subscenarios is pictured in Figure 5.10.

No caching

If we again first compare the cases where caching is not used for both IP and ICN, we see that ICN is able reduce the average, 10th and 90th delay percentile for both backhaul network types compared to IP. The average delay improvement is only 9 ms, while the 90th percentile is lowered by 14 ms. This is in accordance with the plotted kernel density estimate, which for both the IP and ICN cases shows multimodal distributions. The first peak region which is around 8 ms for IP and 7 ms for ICN, can be attributed to all requests/Interests issued for content which is available inside the consumers IoT island.

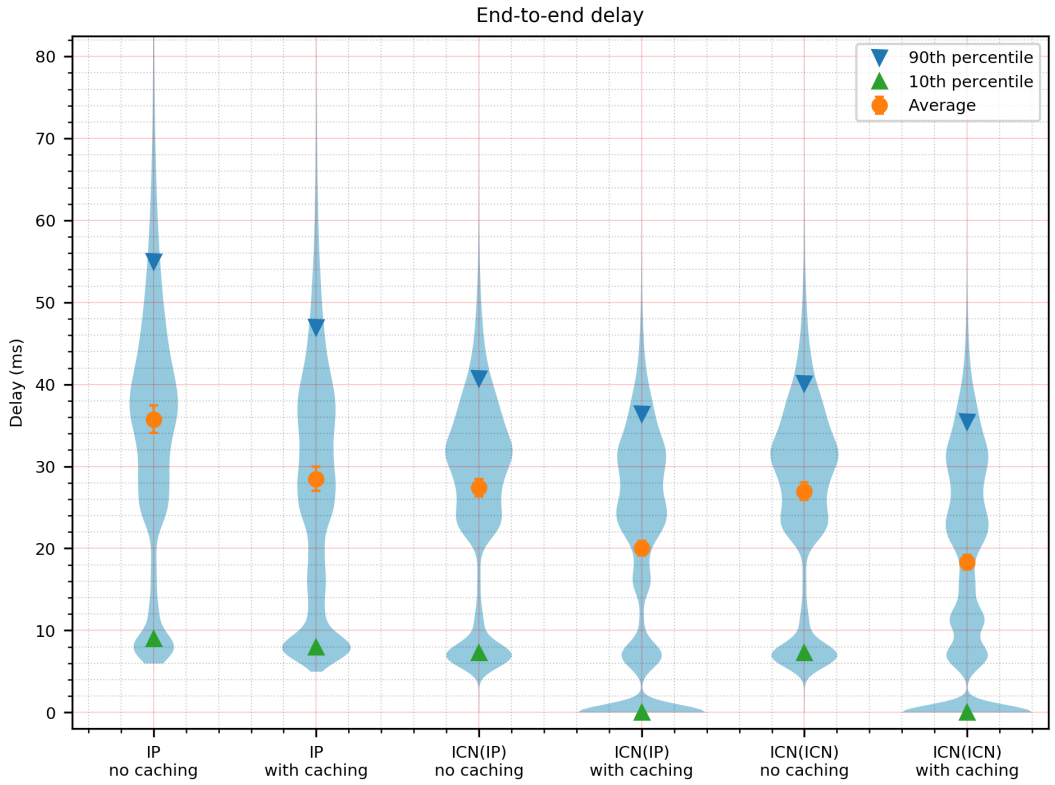


Figure 5.10: Delay results for the 'Smart factory' baseline scenario. The blue shape shows the kernel density estimate (KDE) for the delay distribution. The orange error bars indicate the 95% confidence intervals for the average delay.

It can be concluded that the difference between ICN and IP for this case is minimal. The second region which comprises the average for both IP and ICN, is the result of requests/Interests for data from other IoT islands. The ICN subscenarios show a more compact distribution in this second region, resulting in a lower average delay. This is also reflected by the 90th percentile values. From the ICN subscenarios it can be concluded that having an IP or ICN backhaul, has no significant impact on the end-to-end delay for this 'smart factory' scenario. From this graph we can conclude that the use of ICN does improve the delay for our delay sensitive non-caching subscenarios. This delay reduction is relevant for these delay sensitive 'smart factory' subscenarios without caching, where process optimisation is used to improve production efficiency.

Caching enabled

The IP scenario with CoAP caching enabled, shows a reduced average delay and 90th percentile compared to IP without caching. From the KDE we see that the 'waist' between the upper and lower delay regions, has increased in size compared to the non-caching IP case. This can be attributed to the use of gateway caching. The average and 90th delay percentile are both reduced by 8 ms when CoAP caching is deployed. In contrast to the 'smart home' scenario, we no longer observe 0 ms delay. Consumers reside on the IoT nodes which are not equipped with caches in this IP subscenario, so requests will always have to be transmitted over the 802.15.4 interface.

For the ICN case, we see a similar effect. When ICN enabled IoT islands are deployed with an IP backhaul, we can clearly differentiate the requests/Interests which are satisfied from either the IoT node's own cache (0 ms), from the gateway router cache ($\approx$ 7 ms) or from a remote gateway router ($>$ 16 ms). When the ICN Islands combined with an ICN backhaul network are used, we see an extra peak in the distribution at $\approx$12 ms. This peak must be caused by Interests which are satisfied by a backhaul node. The use of a caching enabled ICN network reduces the average delay by only 9 ms and the 90th delay percentile by 4 ms.

If we compare IP and ICN when caching is enabled, we see that the average delay is reduced by 9-10 ms when ICN is used, depending on ICN's backhaul network. The 90th delay percentile is reduced

by 10-11 ms, depending on the used backhaul network. The 10th percentile of the delay is 0 ms for the ICN subscenarios, because ICN nodes may satisfy Interests from their own cache.

From this graph we can conclude that the use of ICN does also improve the delay for the delay tolerant caching subscenarios. The further improved delay reduction when caching is enabled can be attributed to the ability of IoT nodes, and backhaul nodes to cache data messages. A delay reduction in this order may however not be significant for these delay tolerant caching-enabled subscenarios.

Hop count

The results for this hop count graph are very similar to the delay results since they are strongly correlated. The hop count violin plot for this 'smart factory' scenario is therefore included in appendix Figure C.2.

Network usage

The average network usage for this baseline 'smart factory' scenario is shown in Figure 5.11.

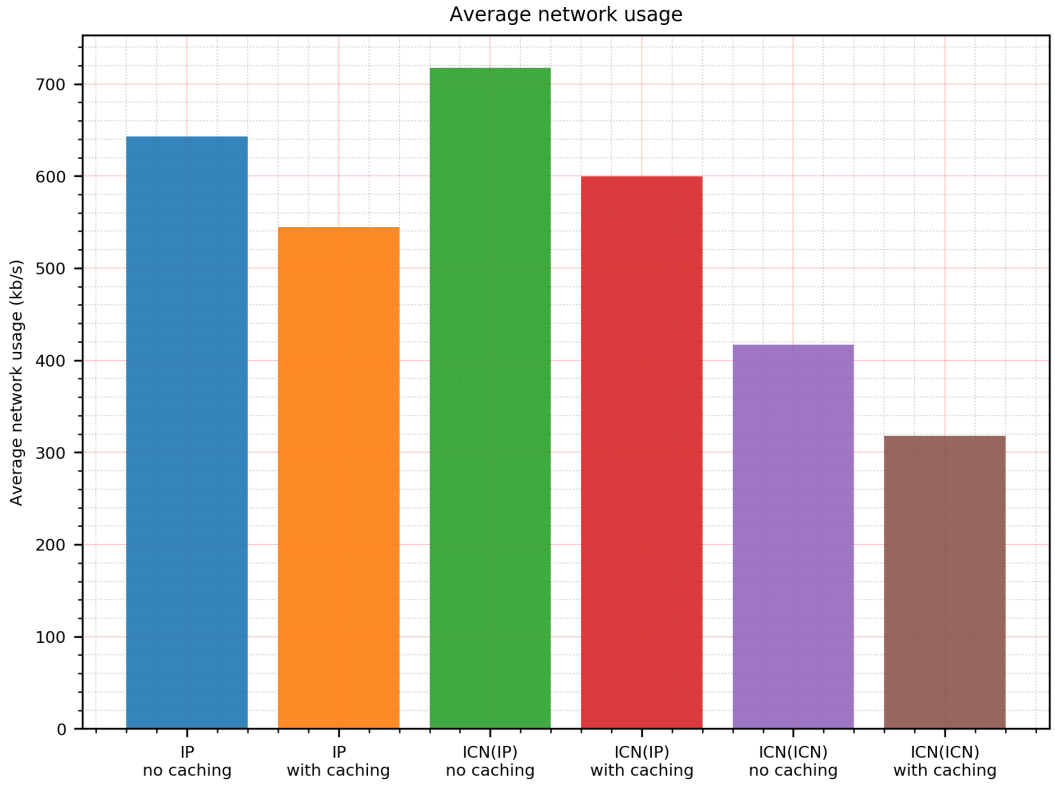


Figure 5.11: Network usage results for the 'Smart factory' baseline scenario.

No caching

From the graph it can be concluded that the IP subscenario without caching does have a lower average network usage than the ICN subscenario with an IP backhaul network. The average network usage is increased for this ICN case by 12%. When IP is compared to ICN with a native ICN backhaul network, a 35% reduction in terms average network usage is achieved.

Caching enabled

If we compare both IP subscenarios, we can conclude that the deployment of IP caching lowers the average network usage by 15%. The ICN network with IP backhaul, is able to reduce the average network usage by 16% when caching is used compare to the ICN(IP) deployment without caching. For the ICN subscenario with ICN backhaul, the largest relative decrease in network usage can be

observed. When caching is enabled, the average network usage is reduced by 24% compared to the ICN(ICN) subscenario without caching.

We can conclude that the average network usage for the ICN (IP) subscenarios is the largest. If we want to reduce the network usage for an IP based IoT deployment, we could use IP caching or ICN with an ICN backhaul network.

Cache hit ratio

Six pie charts in Figure 5.12 show the fraction of requests/Interests which are being satisfied from a cache.

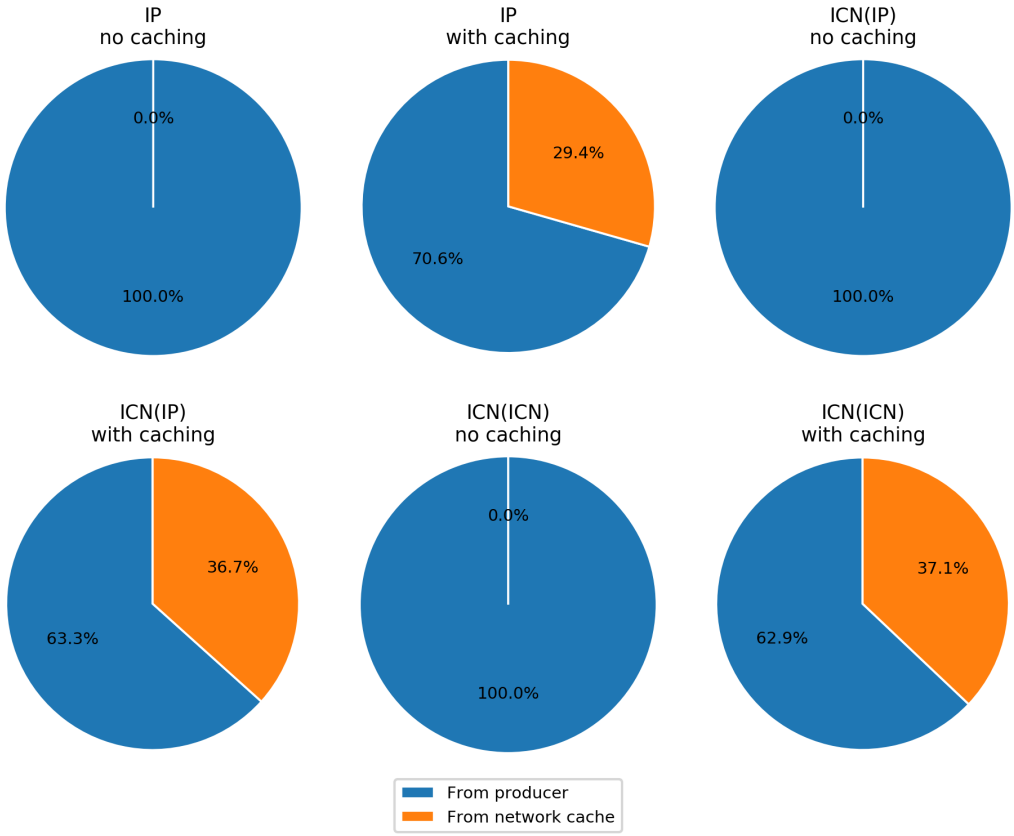


Figure 5.12: Cache hit ratio for the 'Smart factory' baseline scenario. Width of confidence interval is +/- 0.3 %

No caching

The pie charts indicating the cache hit ratio for scenarios where caching is not used again show a trivial result: all requests/Interests are satisfied by a producer.

Caching enabled

If we compare the IP and ICN cache hit ratio, we see that ICN is able to increase the amount of cache hit ratio by respectively 7.3% and 7.7% for the ICN(IP) and ICN(ICN) subscenarios. This effect may be explained by the fact that ICN allows nodes to cache data themselves. When a data message arrives for a neighbouring IoT node, this message is then actively cached. A future Interest from the IoT nodes for this data then may be satisfied from the node's own cache. When caching is also enabled in the backhaul network in the ICN(ICN) subscenario, we see only a minor increase in the number of cache hit ratio. This may be explained by realising that Interests satisfied by the backhaul nodes, may also be satisfied by, for example, a gateway router when caching is disabled in the backhaul. This results in an end-to-end delay decrease but not in a higher cache hit ratio, since the Interest would nevertheless be satisfied from a cache. If we compare these cache hit ratios to the 'smart home' results, we see that

the gain between the cache hit ratio of IP and ICN is increased for this ‘smart factory’ scenario. This can also be explained with the reasoning which is given above. Since the consumers in this scenario reside inside the IoT islands overhearing all transmissions in the IoT island, a cache hit is more likely to occur compared to the CL1 consumers in the ‘Smart home’ scenario.

Cache capacity usage

The cache capacity usage results for this ‘smart factory’ scenario are shown in Figure 5.13. The main conclusion which can be drawn from this graph is that a cache size of 32 kB is sufficient for both ICN as well as for IP supporting IoT networks. The results show that the cache capacity usage never exceeded the maximum capacity. The IP subscenario with CoAP caching enabled used on average just 5,4% of the cache capacity. In our simulations, the maximum cache capacity usage observed was 8.2%. The ICN subscenario with IP backhaul has a similar cache capacity usage for both the gateway router as well as the IoT nodes. This is in accordance with the cache everything strategy, where every transmission in the IoT island is cached by every IoT node including the gateway. We observe a cache capacity usage of 0% at the backhaul nodes, which is caused by IP not being able to cache ICN packets. When we look at the ICN subscenarios with ICN backhaul network, similar cache capacity usages for the gateway routers and IoT nodes can be observed. For this specific case, we additionally see that the backhaul network also caches ICN messages. The shown confidence interval is considerably larger than the other shown intervals. When many shortest paths between the IoT islands cross a specific link, then this link is able to cache a lot of data from multiple islands causing a high cache capacity usage. This effect is highly dependent on the generated topology, causing a wider confidence interval. Nevertheless, we observe that both the maximum and average cache capacity usage are well below the maximum cache capacity, reducing the end-to-end delay as shown in Figure 5.10.

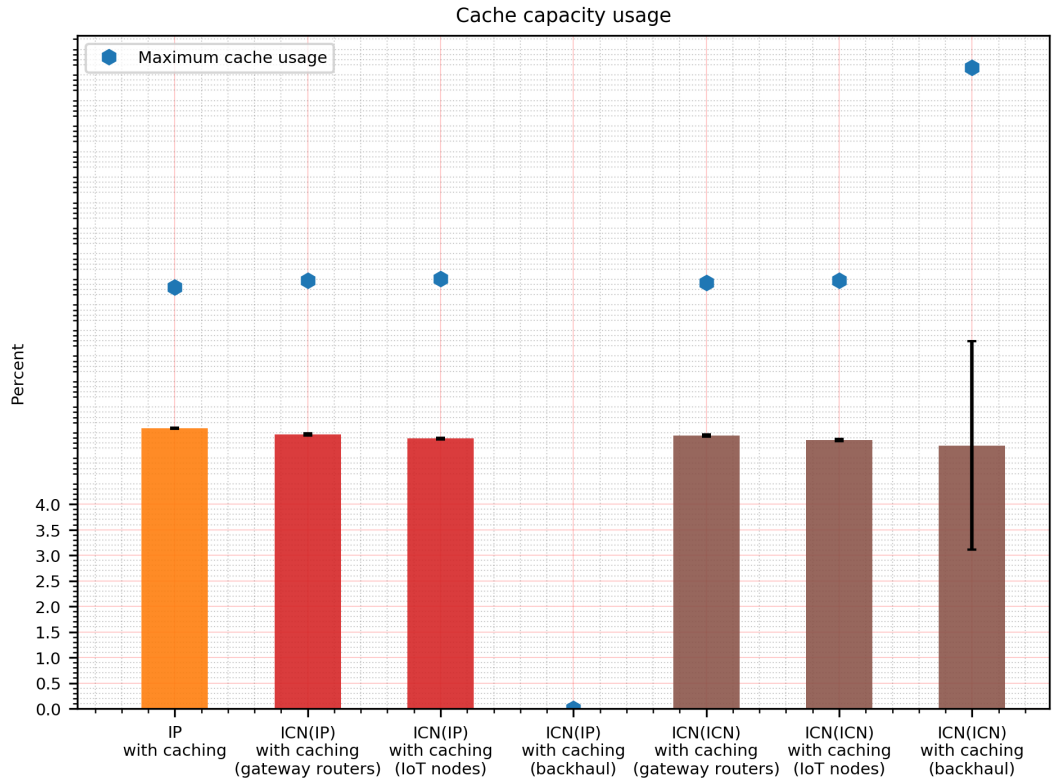


Figure 5.13: Cache capacity usage results for the ‘Smart factory’ baseline scenario. The black error bars indicate the 95% confidence intervals for the average cache capacity usage.

5.2.2. Sensitivity analysis

In this sensitivity analysis, multiple parameters of the baseline scenario are varied, to test its effect on both the IP and ICN cases. In this section we will present the sensitivity study results from the 'Smart factory' scenario. An overview of the covered scenario aspects can be seen in Figure 3.3.

Number of IoT islands

We start by varying the number of IoT nodes per IoT island. We keep the total number of IoT nodes equal to the maximum value for our simulation platform, which is 250 for this scenario. So if we, for example, double the number of IoT Islands, then we half the number of IoT nodes per IoT island. The results can be seen in Figure 5.14. When we look at the end-to-end delay, we see interesting behaviour

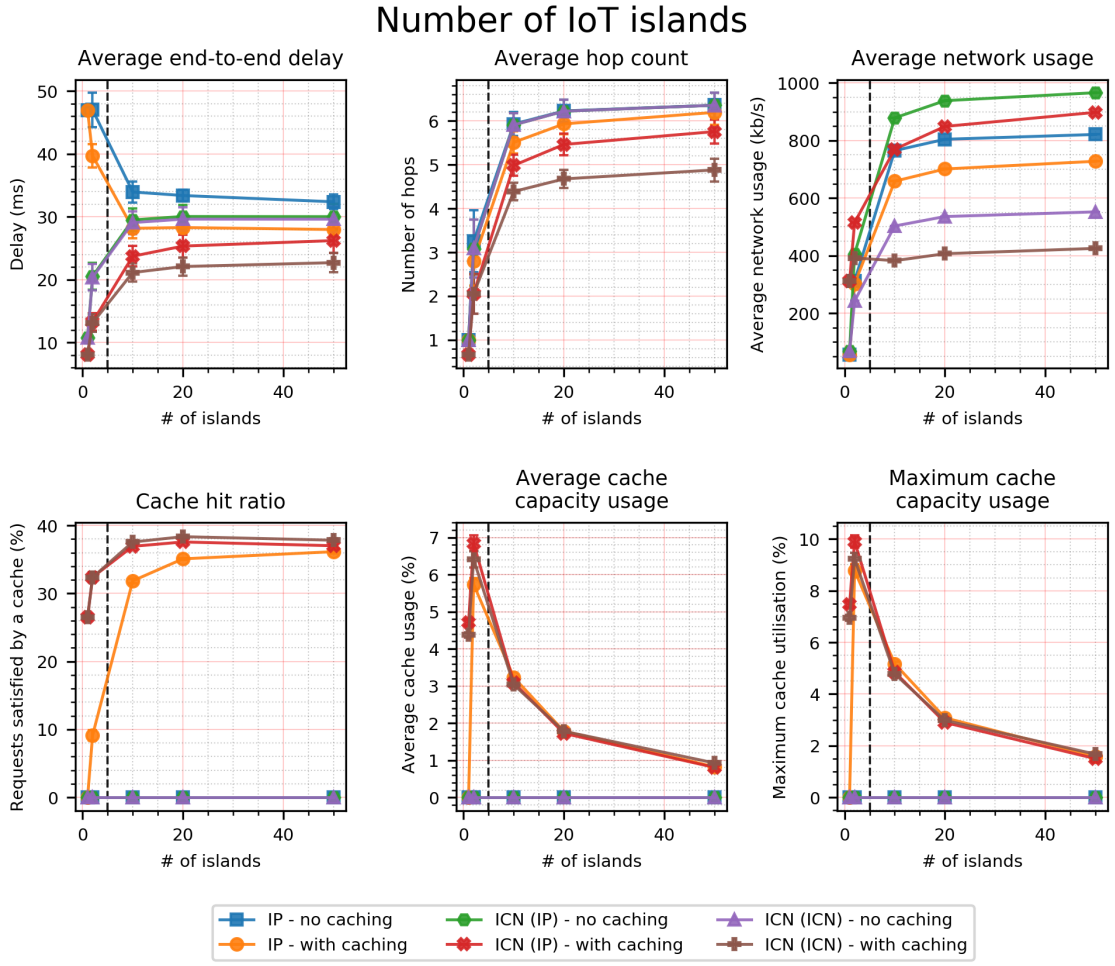


Figure 5.14: Sensitivity study results, when the number IoT islands is changed. The 95% confidence intervals are indicated with error bars. Baseline value for the number of IoT islands is indicated with vertical dotted line.

if we have a small number of islands and thus a large number of nodes per island. The IP subscenarios show a decreasing trend, where the delay starts high and decreases when the number of islands is increased. The ICN subscenarios show a different trend, since the delay starts low and increases when the number of islands is increased. We examined the simulation log files to find a possible explanation for this behaviour. We found that this effect is most likely caused by ICNs stateful forwarding logic. When we have a low number of IoT islands, a high number of IoT nodes are in the same IoT island. There will be a lot of requests/Interests issued, which will be overheard and registered in the PIT of all IoT nodes. Interest aggregation will prevent the transmission of duplicate Interests for the same data. When a consumer wants to send an Interest for specific data which already is requested by another node, it will only result in an updated PIT entry. So less interest transmissions are needed compared to the number of requests in IP. This will result in a lower traffic load with a less congested

wireless channel. IP based deployments do not use a stateful forwarding plane with Interest/request aggregation. Therefore, a lower number of IoT islands with a higher number of IoT nodes will result in a busier transmission channel causing higher end-to-end delays.

An increase in the number of IoT islands typically results in an increase in the number of hops, since the IoT nodes are distributed over multiple islands connected to the backhaul network. This is also clearly shown in the hop count graph. The network usage graph shows a similar ordering as in the baseline scenario.

When we look at the cache hit ratio graph, a large difference is visible between IP and ICN for a low number of IoT islands. When we have a small number of islands, IP caching is providing a lower cache hit ratio compared to ICN because most consumers and producers reside in the same IoT island. The gateway routers cache will only be consulted when requests are issued for producers in other domains. When the number of islands is increased we see that the cache hit ratios start to converge since we then also have a lower number of IoT nodes per island.

The cache capacity usage graphs show that a smaller number of IoT islands will result in a higher cache capacity usage. When the number of IoT nodes per island is high, more requests/Interests will be issued in the same IoT island, also resulting in a higher number of received data messages. These data messages will be received and cached by the IoT nodes, causing a higher cache capacity usage.

Consumer number

The second sensitivity analysis focusses on the number of consumers in the ‘smart factory’ scenario. The results can be seen in Figure 5.15.

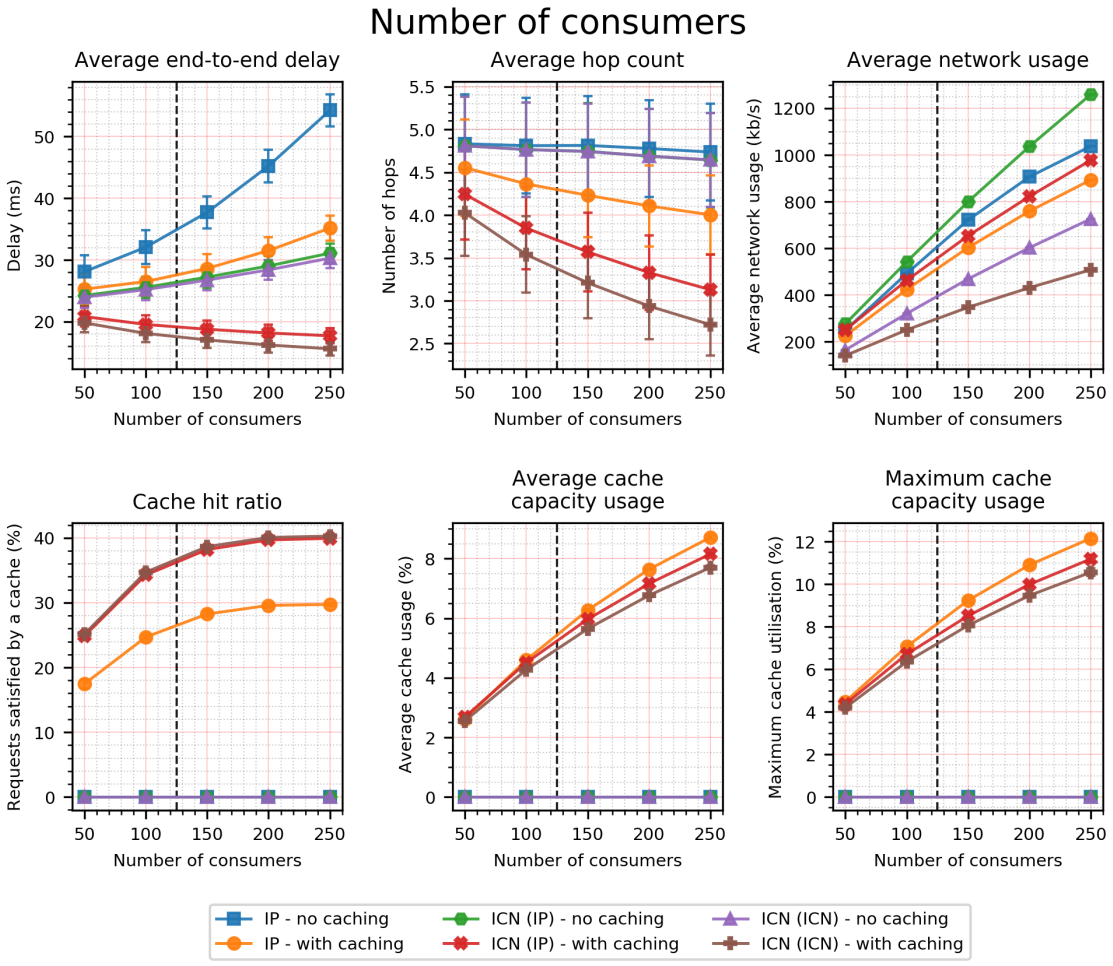


Figure 5.15: Sensitivity study results, when the number of consumers is varied. The 95% confidence intervals are indicated with error bars. Baseline value for the number of consumers is indicated with a dotted vertical line.

We will first look into the dependence of the number of consumers on the end-to-end delay. For the IP subscenario without caching, we see that the delay increases rapidly when the number of consumers is increased. For example, when the number of consumers is doubled, then the observed traffic load will also double since there is no stateful forwarding or caching.

The IP with caching and both ICN subscenarios without caching providing lower end-to-end delays, and the delay increases at a lower rate. When ICN is used with caching enabled, we see an opposite effect, the delay decreases when the number of consumers is increased. A similar effect was observed in the 'smart home' sensitivity analysis. This can be explained by realising that there will be more requests for the same data, allowing nodes to continuously have recent data in cache.

When we look at the average network usage we see again the same ordering as in the baseline scenario. In terms of scalability, it can be said that the ICN(ICN) subscenario with caching uses the network resources most efficiently when the traffic load increases.

From the cache hit ratio graph we see that the ICN subscenarios have a higher cache hit ratio than the IP cases with CoAP caching enabled. The average and maximum cache capacity usage increase when the number of consumers increases. The cache capacity is sufficient for all consumer numbers.

Data freshness period

Data may be cached for a duration specified by the data freshness period. After this time, the cached data may be considered stale and the data should be removed from the cache. We varied the data freshness period to see the effects on our covered key performance indicators. The results can be seen in Figure 5.16.

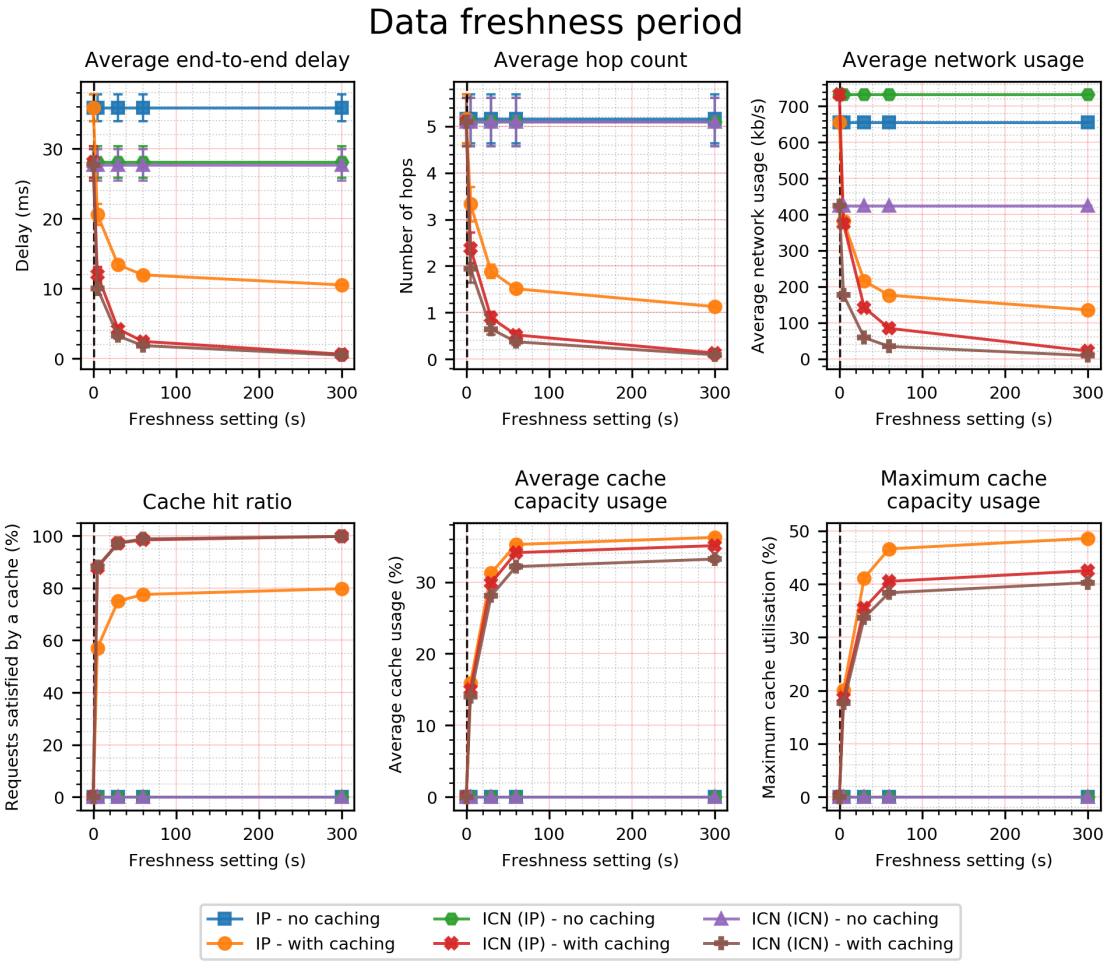


Figure 5.16: Sensitivity study results, when the data freshness duration is changed. The 95% confidence intervals are indicated with error bars. Baseline value for the data freshness period is indicated with a dotted vertical line.

We start this time by looking at the cache hit ratio graph. From this plot it can be seen that a data freshness period of 60 seconds is already enough to satisfy all Interests from a cache for the ICN subscenario and 77% of all requests for the IP subscenario. The IP subscenario will never be able to satisfy all requests from a cache in this 'smart factory' scenario, since requests for a producer in the same IoT island will not cross the gateway. These 23% of requests will therefore be satisfied by the producer itself. It is also interesting to note that even for the ICN scenario where 100% of the data messages is retrieved from a cache, that the cache size of 600 messages is sufficient and will be maximally used for 50%.

If we now look at the end-to-end delay we see a decreasing trend for the IP caching and ICN with caching subscenarios when the data freshness period is increased. As mentioned earlier, a longer data freshness period will result in an improved cache hit ratio. Interests or requests may thus be satisfied by older cached data closer to the consumer. Similar behaviour can be seen in the hop count graph. The network usage graph also shows that the network usage is reduced when more Interests/requests can be satisfied from a cache, reducing the number of required hops.

Content popularity distribution

We now look into the dependence of the selected content popularity distribution on the performance of the covered networking paradigms for this 'smart factory' scenario. We do this by varying the parameter α of the Zipf distribution described in Section 4.2.1. We initially conducted this sensitivity study with the baseline cache size and found that there was no significant effect observable when varying the Zipf parameter for our selected (realistic) values. For this sensitivity study we therefore use the same small cache size of 10 data messages as in the 'smart home' scenario, to make sure that the caches are continuously filled at their full capacity. The results can be found in Figure 5.17.

The end-to-end delay graph outlines that all subscenarios without caching are not significantly influenced by an increase of the Zipf parameter (α). However, the subscenarios where caching is enabled do show a decrease of the end-to-end delay for an increasing α. This can be explained by realising that an increase in α changes the probability mass function of the Zipf distribution since a higher request probability will be assigned to the highest ranked data, making this data relatively more popular. When many IoT nodes request similar data, it is more likely that this data is available in a cache. This is also shown in the cache hit ratio graph, where it is shown that an increase in α allows a higher number of requests to be satisfied from a cache. The cache capacity usage graphs show that the caches use their full capacity for all covered popularity distributions.

The average network usage graph shows a decrease of the the network usage when the Zipf parameter is increased. More requests can then be satisfied from caches, requiring fewer transmissions and thereby lowering the network usage.

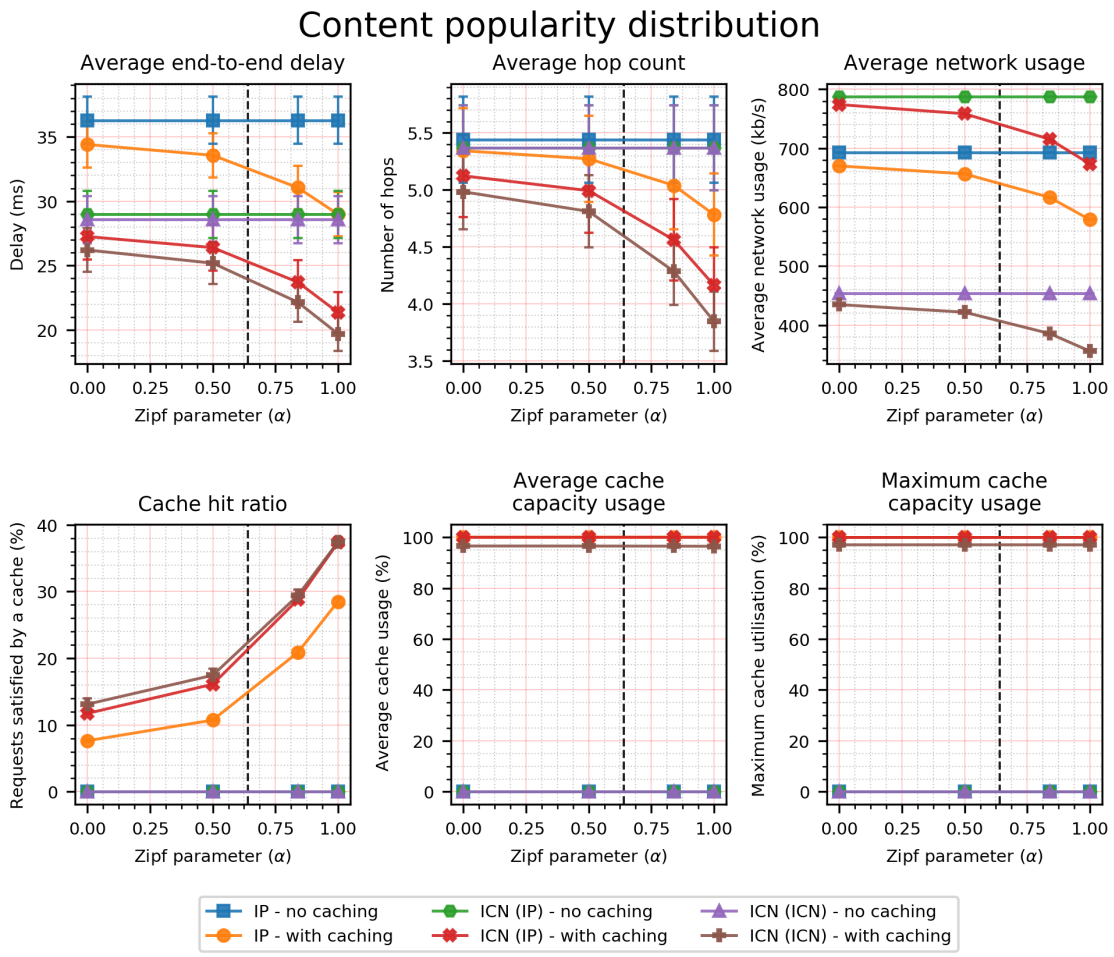


Figure 5.17: Sensitivity study results, when the content popularity distribution is changed. The 95% confidence intervals are indicated with error bars. Baseline value for Zipf parameter (α) is indicated with vertical dotted line.

Cache size

The last sensitivity study done for this 'smart factory' scenario will cover multiple settings for the cache size. The results can be seen in Figure 5.18. From Figure 5.18 we can determine how large the caches should be for our 'smart factory' scenario. From the cache hit graph we see that an increase in cache size from 100 to 1000 data messages does not result in a higher number of requests satisfied by caching for both IP and ICN. Similar behaviour can be seen in the end-to-end delay, hop count and average network usage graph where a cache size of 1000 messages does not change the results compared to a cache size of 100 messages. It can therefore be concluded that the minimal cache size to be able to fully exploit the benefits of caching will be 100 messages.

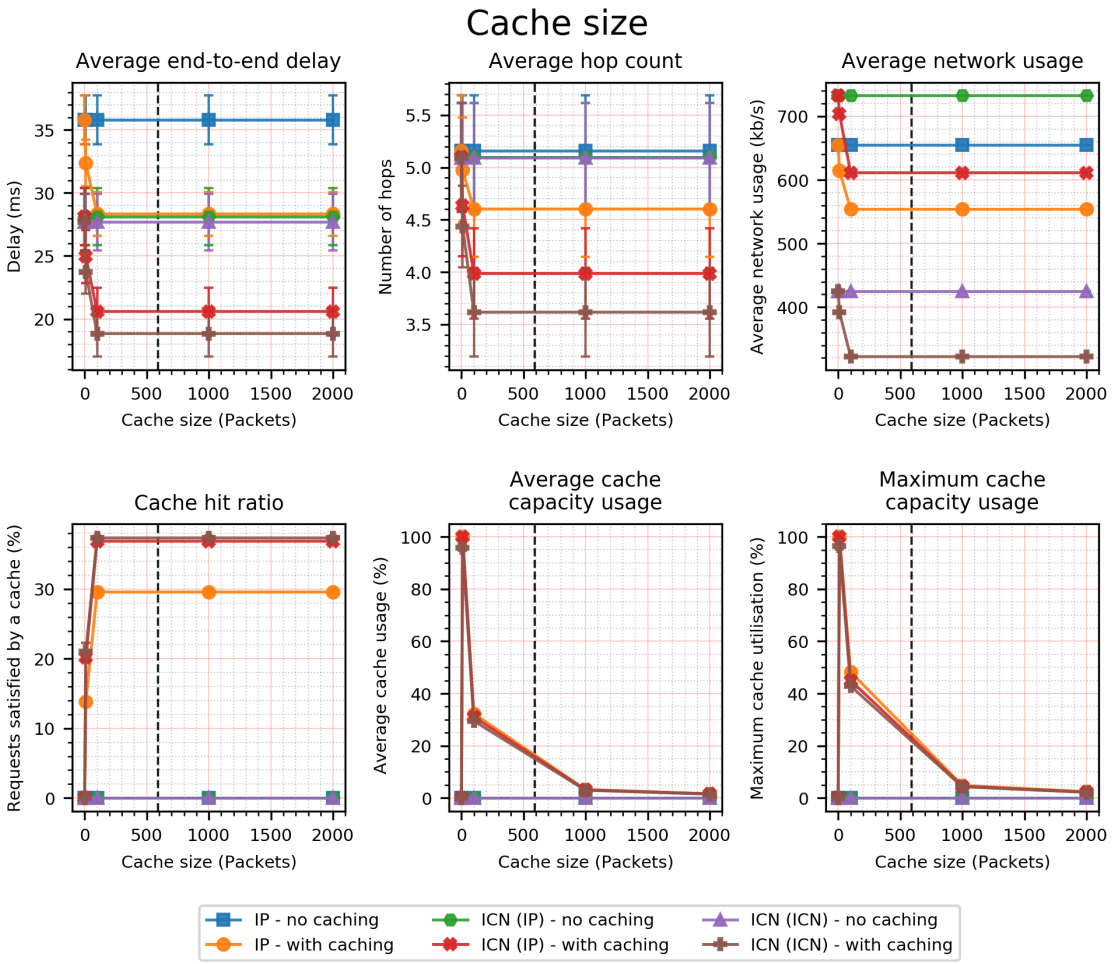


Figure 5.18: Sensitivity study results, when the cache size is changed. The 95% confidence intervals are indicated with error bars. Baseline value for the cache size is indicated with vertical dotted line.

6

Conclusion and future work

6.1. Conclusion

In this thesis we have presented a comparison of IP and ICN for IoT applications. Our main research goal is to asses whether ICN is advantageous for the IoT, by comparing an ICN approach for IoT to an IP approach for IoT applications. We started by defining two relevant IoT scenarios based on the McKinsey report which describes the most promising IoT use cases for the future [12]. We selected a 'smart home' scenario and a 'smart factory' scenario, because these scenarios offered the opportunity to cover most of our scenario aspects, either in the baseline scenario or in the sensitivity study. For the selection of comparison metrics we first identified the two main advantages of ICN for IoT applications proposed in literature: ICN should allow a more efficient use of network resources and deliver an improved content delivery performance. Especially the more efficient use of network resources is an important feature for IoT applications, since it is expected that a very large number of IoT devices will be connected to the internet. These expected gains of ICN are due to use of a stateful forwarding plane, which additionally enables the use of caching at the network layer. To allow for a fair comparison between ICN and IP we therefore also compared ICN to IP deployments with application level caching. Moreover, we compared IP to ICN deployments with two types of backhaul networks interconnecting the IoT islands. We covered an ICN-aware backhaul, capable of interpreting and caching ICN messages. Additionally, we covered ICN deployments with an IP based backhaul network. This ICN(IP) subscenario is considered to be the most realistic short-term ICN deployment which can be easily connected to the current internet. Five metrics were selected, which were used to verify these claims. The improved network efficiency was examined based on the network usage meric and the improved content delivery performance was examined with the end-to-end delay. A special cache hit ratio metric combined with the cache capacity usage metric allowed us to asses the advantages of having caching in IP and ICN deployments. The NS-3 based ndnSIM simulator was used to perform the experiments, since it offered the most well-rounded evaluation platform.

In the baseline 'smart home' scenario we have seen that ICN is able to provide a small end-to-end delay reduction compared to IP, for both the caching and non-caching subscenarios. This delay improvement is very small and not significant for monitoring applications such as in 'smart homes'. The most realistic short term application of ICN where it is used on top of IP does not result in a more efficient use of network resources. Compared to IP the average network usage is increased, when ICN is combined with IP. ICN can only lower the average network usage when an ICN backhaul is used for this 'smart home' scenario.

From the sensitivity study we can conclude that consumers in the backhaul contribute the most to the average network usage, especially when ICN with IP backhaul is deployed. When the total number of consumers increases we see that an ICN(ICN) deployment leads to an improved delay performance and a reduced network usage. Moreover, we identified that the data freshness period setting and the content popularity distribution have the strongest impact on the ICN(ICN) scenario with caching enabled. The IP caching and ICN(IP) deployment with caching have a similar dependency on both scenario aspects. From the cache size sensitivity study, we were able to conclude that a very small cache which can store at least ten packets is already sufficient for the 'smart home' scenario.

For the ‘smart factory’ scenario we can conclude that the use of ICN does improve the delay by 9-10 ms for both the caching as well as the non-caching subscenarios compared to IP for the baseline ‘smart factory’ scenario. This translates to a relative delay reduction of about 25% if ICN and IP are compared with caching disabled. This average delay reduction is relevant for the delay sensitive ‘smart factory’ subscenarios without caching, where process optimisation is used to improve production efficiency. When we compare the delay tolerant subscenarios where caching is enabled for both ICN and IP we see a relative delay reduction of about 35% when ICN is used. The average network usage is again the highest for the ICN(IP) subscenarios. The largest reduction in network usage of 41% will occur when ICN is deployed with an ICN backhaul instead of an IP-based deployment. From the sensitivity analysis we can conclude that ICNs stateful forwarding plane helps to reduce the average end-to-end delay about a factor six compared to IP when large scale wireless IoT islands are used. Moreover, when the number of consumers is increased, the ICN deployments provide the lowest end-to-end delay. When we look at the data freshness period setting and the content popularity distribution, we see that the relative performance between our covered subscenarios does not change significantly. We also found that a cache which can hold 100 data messages is already sufficient for the ‘smart factory’ baseline scenario.

The following main conclusions can be drawn.

- ICN is able to reduce the end-to-end delay for all subscenarios. This delay reduction is however not overly significant for delay tolerant applications, such as our ‘smart home’ scenario and the ‘smart factory’ scenarios with caching enabled. The delay improvement is relevant for our covered delay sensitive ‘smart factory’ subscenarios.

- Deploying ICN with an ICN backhaul instead of IP does reduce the network usage for all subscenarios. This is an important benefit since, as mentioned earlier, it is expected that the total number of connected IoT device will massively increase in the coming years and a resource efficient solution is needed to deal with this traffic increase.

- The realistic deployment of ICN with an IP backhaul will result in an increase of network usage for all covered subscenarios, compared to a classical IP deployment. If an end-to-end delay reduction is not needed, it is better to keep using IP.

- ICN seems to be better suited for scenarios with larger number of users and consumers, as we saw in our sensitivity study. The use of ICN in small size scenarios such as, for example, farming scenarios with a low number of consumers may benefit less from an ICN deployment, compared to a large scale applications such as smart cities with a large number of users and interested consumers.

6.2. Future work

Future work should focus on covering other IoT scenarios, for example, scenarios where different physical layer technologies are used, different topologies such as grid-based IoT networks and scenarios with strict and explicitly known delay requirements.

Our developed simulation model can also be used in the future to compare Cisco’s hybrid ICN (hICN) implementation [91] to other ICN-over-IP solutions. This new technology may remove the drawbacks of having a higher network usage when ICN in transported over IP networks, since standard IP packets are used as ICN messages.

Furthermore, future work may also focus on comparing IoT IP and ICN scenarios which include mobile nodes. This could be an interesting research direction due to the native consumer mobility support feature of ICN.

Bibliography

[1] V. Jacobson, “Congestion avoidance and control,” *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, Aug. 1988.

[2] V. Jacobson, “Google Tech Talks: A New Way to Look at Networking,” https://www.youtube.com/watch?v=oCZMoY3q2uM.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[4] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, “Adaptive Forwarding in Named Data Networking ,” *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, Jun. 2012.

[5] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. C. Schmidt, and M. Wählisch, “Information-Centric Networking (ICN) Research Challenges,” RFC 7927, Jul. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7927.txt

[6] “Named Data Networking ,” https://named-data.net/.

[7] “CCNx project,” https://blogs.parc.com/ccnx/.

[8] “Cisco Acquires PARC’s Content Centric Networking (CCN) Platform,” http://blogs.parc.com/2017/02/cisco-acquires-parcs-content-centric-networking-ccn-platform/.

[9] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[10] “Gartner, Inc,” https://www.gartner.com/smarterwithgartner/, 2017.

[11] S. Lucero *et al.*, “Iot platforms: enabling the internet of things,” *HIS Technology. Retrieved from https://cdn. ihs. com/www/pdf/enabling-IOT. pdf*, 2016.

[12] M. James, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, “The Internet of Things: Mapping the value beyond the hype,” *McKinsey Global Institute*, p. 3, 2015.

[13] O. Vermesan and P. Friess, *Internet of Things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[14] “IEEE 802.15 Working Group for Wireless Personal Area Networks,” http://ieee802.org/15/index.html.

[15] “Bluetooth LE,” https://www.bluetooth.com/specifications.

[16] “3GPP Release 13,” http://www.3gpp.org/release-13.

[17] “LoRaWAN,” https://www.lora-alliance.org/.

[18] “Sigfox,” https://www.sigfox.com/en.

[19] M. Amadeo, C. Campolo, A. Molinaro, and G. Ruggeri, “Content-centric wireless networking: A survey,” *Computer Networks*, vol. 72, pp. 1 – 13, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614002497

[20] S. Sun, L. Han, and S. Han, “GRMR: greedy regional multicast routing for wireless sensor networks,” *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 1, pp. 21–29, 2016.

[21] S. Gao, H. Zhang, and B. Zhang, “Energy Efficient Interest Forwarding in NDN-Based Wireless Sensor Networks,” *Mobile Information Systems*, vol. 2016, 2016.

[22] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, “Information centric networking in the IoT: Experiments with NDN in the wild,” in *Proceedings of the 1st International Conference on Information-centric Networking*. ACM, 2014, pp. 77–86.

[23] M. A. M. Hail, M. Amadeo, A. Molinaro, and S. Fischer, “On the performance of caching and forwarding in information-centric networking for the IoT,” in *International Conference on Wired/Wireless Internet Communication*. Springer, 2015, pp. 313–326.

[24] S. K. Datta and C. Bonnet, “Integrating Named Data Networking in Internet of Things architecture,” in *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. IEEE, 2016, pp. 1–2.

[25] O. Waltari and J. Kangasharju, “Content-Centric Networking in the Internet of Things,” in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2016, pp. 73–78.

[26] S. Li, Y. Zhang, D. Raychaudhuri, R. Ravindran, Q. Zheng, L. Dong, and G. Wang, “IoT middleware architecture over Information-Centric Networks,” in *Globecom Workshops (GC Wkshps), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[27] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, “Information centric networking in IoT scenarios: The case of a smart home,” in *2015 IEEE International Conference on communications (ICC)*. IEEE, 2015, pp. 648–653.

[28] Z. Li, J.-C. Point, S. Ciftci, O. Eker, G. Mauri, M. Savi, and G. Verticale, “ICN based shared caching in future converged fixed and mobile network,” in *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2015, pp. 1–6.

[29] A. El-Mougy, M. Ibnkahla, and L. Hegazy, “Software-defined wireless network architectures for the Internet-of-Things,” in *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, Oct 2015, pp. 804–811.

[30] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, “Sensor search techniques for sensing as a service architecture for the Internet of Things,” *IEEE Sensors Journal*, vol. 14, no. 2, pp. 406–420, Feb. 2014.

[31] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, “Caching in Named Data Networking for the wireless Internet of Things,” in *2015 International Conference on Recent Advances in Internet of Things (RIoT)*. IEEE, 2015, pp. 1–6.

[32] O. Hahm, E. Baccelli, T. C. Schmidt, M. Wahlisch, and C. Adjih, “A named data network approach to energy efficiency in IoT,” in *Globecom Workshops (GC Wkshps), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[33] A. Lindgren, F. B. Abdesslem, B. Ahlgren, O. Schelén, and A. M. Malik, “Design choices for the IoT in information-centric networks,” in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 882–888.

[34] A. Compagno, M. Conti, and R. Droms, “OnboardICNg: A secure protocol for on-boarding IoT devices in ICN,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’16. New York, NY, USA: ACM, 2016, pp. 166–175.

[35] M. Enguehard, R. Droms, and D. Rossi, “On the Cost of Secure Association of Information Centric Things,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’16. New York, NY, USA: ACM, 2016, pp. 207–208.

[36] J. Suarez, J. Quevedo, I. Vidal, D. Corujo, J. Garcia-Reinoso, and R. L. Aguiar, “A secure IoT management architecture based on Information-Centric Networking,” *Journal of Network and Computer Applications*, vol. 63, pp. 190 – 204, 2016.

[37] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Securing instrumented environments over content-centric networking: The case of lighting control and NDN,” in *IEEE INFOCOM NOMEN Workshop*, 2013.

[38] A. M. Malik, J. Borgh, and B. Ohlman, “Attribute-Based Encryption on a Resource Constrained Sensor in an Information-Centric Network,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’16. New York, NY, USA: ACM, 2016, pp. 217–218.

[39] . Zhang, A. Afanasyev, J. Burke, and L. Zhang, “A survey of mobility support in Named Data Networking,” in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2016, pp. 83–88.

[40] J. Quevedo, D. Corujo, and R. Aguiar, “A case for ICN usage in IoT environments,” in *2014 IEEE Global Communications Conference*, Dec 2014, pp. 2770–2775.

[41] P. Zuraniewski, N. van Adrichem, D. Ravesteijn, W. IJntema, C. Papadopoulos, and C. Fan, “Facilitating ICN deployment with an extended openflow protocol,” in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 123–133.

[42] “Thread network protocol,” http://threadgroup.org/About.

[43] “WirelessHART communication protocol,” https://fieldcommgroup.org/.

[44] “Zigbee Alliance,” http://www.zigbee.org/.

[45] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” RFC 4944, Sep. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4944.txt

[46] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” United States, 1998.

[47] “Extensible Messaging and Presence Protocol (XMPP),” https://xmpp.org/.

[48] “MQTT Version 3.1.1 Plus Errata 01 specification,” http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[49] J. O’Hara, “Toward a commodity enterprise middleware,” *Queue*, vol. 5, no. 4, pp. 48–55, May 2007.

[50] “REST Architectures,” https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest.

[51] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[52] V. Karagiannis, P. Chatzimisios, F. Vázquez-Gallego, and J. Alonso-Zarate, “A survey on application layer protocols for the Internet of Things,” vol. 3, pp. 11–17, 01 2015.

[53] C. Gündoğan, T. C. Schmidt, and M. Wählisch, “Publish-Subscribe Deployment Option for NDN in the Constrained Internet of Things,” Internet Engineering Task Force, Internet-Draft draft-gundogan-icnrg-pub-iot-01, Jul. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-gundogan-icnrg-pub-iot-01

[54] M. Laner, P. Svoboda, N. Nikaein, and M. Rupp, “Traffic models for machine type communications,” in *Proceedings of the Tenth International Symposium on Wireless Communication Systems (ISWCS 2013)*. VDE, 2013, pp. 1–5.

[55] O. Al-Khatib, W. Hardjawana, and B. Vucetic, “Traffic modeling for Machine-to-Machine (M2M) last mile wireless access networks,” in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 1199–1204.

[56] M. A. Mehaseb, Y. Gadallah, and H. El-Hennawy, “Wsn application traffic characterization for integration within the Internet of Things,” in *2013 IEEE Ninth International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE, 2013, pp. 318–323.

[57] K. Pentikousis, B. Ohlman, E. B. Davies, S. Spirou, and G. Boggia, “Information-Centric Networking: Evaluation and Security Considerations,” RFC 7945, Sep. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7945.txt

[58] “NXP JN5168: ZigBee and IEEE802.15.4 wireless microcontroller,” https://www.nxp.com/products/wireless-connectivity/zigbee/zigbee-and-ieee802.15.4-wireless-microcontroller-with-256-kb-flash-32-kb-ram:JN5168.

[59] K. Mikhaylov, J. Tervonen, J. Heikkilä, and J. Känsäkoski, “Wireless sensor networks in industrial environment: Real-life evaluation results,” in *2012 2nd Baltic Congress on Future Internet Communications (BCFIC)*. IEEE, 2012, pp. 1–7.

[60] A.-S. Tonneau, N. Mitton, and J. Vandaele, “How to choose an experimentation platform for wireless sensor networks? a survey on static and mobile wireless sensor network experimentation facilities,” *Ad Hoc Networks*, vol. 30, pp. 115 – 127, 2015.

[61] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, “Internet of Things (IoT): Research, simulators, and testbeds,” *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[62] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, “FIT IoT-LAB: A large scale open experimental IoT testbed,” in *Proc. IEEE 2nd World Forum Internet of Things (WF-IoT)*, Dec. 2015, pp. 459–464.

[63] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, “The w-iLab.t Testbed,” in *Testbeds and Research Infrastructures. Development of Networks and Communities*, T. Magedanz, A. Gavras, N. H. Thanh, and J. S. Chase, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–154.

[64] K. Wehrle, J. Reber, and V. Kahmann, “A simulation suite for internet nodes with the ability to integrate arbitrary quality of service behavior,” in *Proceedings of Communication Networks And Distributed Systems Modeling And Simulation Conference (CNDS 2001), Phoenix (AZ)*, 2001.

[65] “CCN-lite Project,” http://www.ccn-lite.net.

[66] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki - a lightweight and flexible operating system for tiny networked sensors,” in *Proc. 29th Annual IEEE Int. Conf. Local Computer Networks*, Nov. 2004, pp. 455–462.

[67] “MiniNDN,” https://github.com/named-data/mini-ndn.

[68] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 19:1–19:6.

[69] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, “NDNSIM 2: An updated NDN simulator for NS-3,” Technical Report NDN-0028, Revision 2, NDN, Tech. Rep., 2016.

[70] A. Afanasyev, I. Moiseenko, L. Zhang *et al.*, “NDNSIM: NDN simulator for NS-3,” *University of California, Los Angeles, Tech. Rep*, vol. 4, 2012.

[71] H. Tazaki, F. Urbani, and T. Turletti, “DCE cradle: simulate network protocols with real stacks for better realism,” in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 153–158.

[72] L. Saino, I. Psaras, and G. Pavlou, “Icarus: a caching simulator for Information Centric Networking (ICN),” in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS ’14. ICST, Brussels, Belgium, Belgium: ICST, 2014.

[73] “NS-3 Low-Rate Wireless Personal Area Network (LR-WPAN) module,” https://www.nsnam.org/docs/models/html/lr-wpan.html.

[74] V. Rege and T. Pecorella, “A Realistic MAC and Energy Model for 802.15. 4,” in *Proceedings of the Workshop on ns-3*. ACM, 2016, pp. 79–84.

[75] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: Universal Topology Generation from a User”s Perspective,” Boston, MA, USA, Tech. Rep., 2001.

[76] P. V. Mieghem, *Data Communications Networking*. West Lafayette, IN, USA: Purdue University Press, 2006.

[77] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[78] B. M. Waxman, “Routing of multipoint connections,” *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[79] M. Naldi, “Connectivity of waxman topology models,” *Computer communications*, vol. 29, no. 1, pp. 24–31, 2005.

[80] M. Roughan, J. Tuke, and E. Parsonage, “Estimating the parameters of the waxman random graph,” *arXiv preprint arXiv:1506.07974*, 2015.

[81] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet Topology Zoo,” *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[82] S. Bar, M. Gonen, and A. Wool, “An incremental super-linear preferential internet topology model,” in *Passive and Active Network Measurement*, C. Barakat and I. Pratt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 53–62.

[83] “NS-3 6LoWPAN module: Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” https://www.nsnam.org/docs/models/html/sixlowpan.html.

[84] “NS-3 IPv6 module,” https://www.nsnam.org/docs/models/html/ipv6.html.

[85] “NS-3 RIPnG module,” https://www.nsnam.org/doxygen/classns3_1_1_rip_ng.html.

[86] “NS-3 UDP Echo server example script,” https://www.nsnam.org/doxygen/udp-echo-server_8cc.html.

[87] “NS-3 UDP Echo client example script,” https://www.nsnam.org/doxygen/udp-echo-client_8cc.html.

[88] “ndnSIM: Consumerzipfmandelbrot application,” http://ndnsim.net/1.0/doxygen/ndn-consumer-zipf-mandelbrot_8cc_source.html.

[89] “ndnSIM: Producer application,” https://ndnsim.net/2.1/doxygen/classns3_1_1ndn_1_1Producer.html.

[90] PARC, “CCNx over UDP,” https://datatracker.ietf.org/doc/slides-interim-2015-icnrg-3-15/.

[91] Cisco, “Mobile Video Delivery with Hybrid ICN,” https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/ultra-services-platform/mwc17-hicn-video-wp.pdf, 2017.

[92] “Parsing scripts GitHub repository,” https://github.com/FBDr/sixparse.

[93] “GitHub ICN for IoT repository,” https://github.com/FBDr/6lowpanstars.

[94] “Violin plot: Kernel-density estimate using Gaussian kernels,” https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.violinplot.html.

[95] O. Hahm, C. Adjih, E. Baccelli, T. C. Schmidt, and M. Wählisch, “ICN over TSCH: Potentials for Link-Layer Adaptation in the IoT,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’16. New York, NY, USA: ACM, 2016, pp. 195–196.

[96] L. M. J.S.M., V. Lokesh, and G. C. Polyzos, “Energy efficient context based forwarding strategy in Named Data Networking of things,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’16. New York, NY, USA: ACM, 2016, pp. 249–254.

[97] B. Ahlgren, A. Lindgren, and Y. Wu, “Demo: Experimental Feasibility Study of CCN-lite on Contiki Motes for IoT Data Streams,” in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’16. New York, NY, USA: ACM, 2016, pp. 221–222.

[98] E. Borgia, R. Bruno, M. Conti, D. Mascitti, and A. Passarella, “Mobile edge clouds for Information-Centric IoT services,” in *2016 IEEE Symposium on Computers and Communication (ISCC)*, June 2016, pp. 422–428.

[99] L. Dong and G. Wang, “Support context-aware IoT content request in Information Centric networks,” in *2016 25th Wireless and Optical Communication Conference (WOCC)*, May 2016, pp. 1–4.

[100] Y. Ye, Y. Qiao, B. Lee, and N. Murray, “PIoT: Programmable IoT using Information Centric Networking,” in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 825–829.

[101] L. Dong, R. Ravindran, and G. Wang, “ICN based distributed IoT resource discovery and routing,” in *2016 23rd International Conference on Telecommunications (ICT)*, May 2016, pp. 1–7.

[102] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, “Named Data Networking of Things (Invited Paper),” in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, April 2016, pp. 117–128.

[103] S. CC, V. Raychoudhury, G. Marfia, and A. Singla, “A survey of routing and data dissemination in delay tolerant networks,” *J. Netw. Comput. Appl.*, vol. 67, no. C, pp. 128–146, May 2016.

[104] Q. Zheng, Q. Li, A. Azgin, and S. O. Amin, “Hey, i know you are moving: producer movement status privacy in information-centric networking,” in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2016, pp. 1080–1085.

[105] C. Moon, S. Han, H. Woo, and D. Kim, “Named Data Networking for infrastructure wireless networks,” in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2016, pp. 343–344.

[106] 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*, vol. 30, no. 2, pp. 92–100, March 2016.

[107] Z. Zhang, H. Ma, and L. Liu, “Cache-Aware Named-Data Forwarding in Internet of Things,” in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[108] M. A. Hail and S. Fischer, “IoT for AAL: An Architecture via Information-Centric Networking,” in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec 2015, pp. 1–6.

[109] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, “Cache coherence in Machine-to-Machine Information Centric Networks,” in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Oct 2015, pp. 430–433.

[110] C. Seales, T. Do, E. Belyi, and S. Kumar, “PHINet: A Plug-n-Play Content-centric Testbed Framework for Health-Internet of Things,” in *2015 IEEE International Conference on Mobile Services*, June 2015, pp. 368–375.

[111] J. Quevedo, M. Antunes, D. Corujo, D. Gomes, and R. L. Aguiar, “On the Application of Contextual IoT Service Discovery in Information Centric Networks,” *Comput. Commun.*, vol. 89, no. C, pp. 117–127, Sep. 2016.

[112] S. Signorello, R. State, and O. Festor, “Exploring IoT Protocols Through the Information-Centric Networking’s Lens,” in *Intelligent Mechanisms for Network Configuration and Security*, S. Latré, M. Charalambides, J. François, C. Schmitt, and B. Stiller, Eds. Cham: Springer International Publishing, 2015, pp. 56–60.

[113] R. Li and H. Asaeda, “Community-oriented networking technology,” *Journal of the National Institute of Information and Communications Technology*, vol. 62, no. 2, pp. 133–138, 2015, cited By 0.

[114] X. Xu, T. Jiang, L. Pu, T. Qiu, and Y. Hu, “A comparison study of connected vehicle systems between Named Data Networking and IP,” *Journal of Internet Technology*, vol. 16, no. 2, pp. 343–350, 2015, cited By 0.

[115] N. Fotiou, K. Katsaros, G. Xylomenos, and G. Polyzos, “H-pastry: An inter-domain topology aware overlay for the support of name-resolution services in the future internet,” *Computer Communications*, vol. 62, pp. 13–22, 2015, cited By 6.

[116] G. C. Polyzos and N. Fotiou, “Building a reliable Internet of Things using Information-Centric Networking,” *Journal of Reliable Intelligent Environments*, vol. 1, no. 1, pp. 47–58, Jul 2015.

[117] S. Li, Y. Zhang, D. Raychaudhuri, and R. Ravindran, “A comparative study of MobilityFirst and NDN based ICN-IoT architectures,” in *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Aug 2014, pp. 158–163.

[118] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, “Secure sensing over Named Data Networking,” in *2014 IEEE 13th International Symposium on Network Computing and Applications*, Aug 2014, pp. 175–180.

[119] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, “Named Data Networking for IoT: An architectural perspective,” in *2014 European Conference on Networks and Communications (EuCNC)*, June 2014, pp. 1–5.

[120] Z. Yan, S. Zeadally, and Y. J. Park, “A Novel Vehicular Information Network Architecture Based on Named Data Networking (NDN),” *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 525–532, Dec 2014.

[121] M. Amadeo, C. Campolo, and A. Molinaro, “Internet of Things via Named Data Networking : The support of push traffic,” in *2014 International Conference and Workshop on the Network of the Future (NOF)*, Dec 2014, pp. 1–5.

[122] H. Yue, L. Guo, R. Li, H. Asaeda, and Y. Fang, “Dataclouds: Enabling community-based data-centric services over the Internet of Things,” *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 472–482, Oct 2014.

[123] M. Amadeo, C. Campolo, and A. Molinaro, “Multi-source data retrieval in IoT via Named Data Networking,” in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN ’14. New York, NY, USA: ACM, 2014, pp. 67–76.

[124] J. Quevedo, D. Corujo, and R. Aguiar, “Consumer driven information freshness approach for content centric networking,” in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2014, pp. 482–487.

[125] O. Briante, M. Amadeo, C. Campolo, A. Molinaro, S. Y. Paratore, and G. Ruggeri, “edomus: User-home interactions through facebook and Named Data Networking,” in *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2014, pp. 155–157.

[126] D. Corujo, R. L. Aguiar, I. Vidal, J. Garcia-Reinoso, and K. Pentikousis, “Research challenges towards a managed information-centric network of things,” in *2014 European Conference on Networks and Communications (EuCNC)*, June 2014, pp. 1–5.

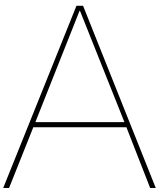
[127] R. Ravindran, X. Liu, A. Chakraborti, X. Zhang, and G. Wang, “Towards software defined ICN based edge-cloud services,” in *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, Nov 2013, pp. 227–235.

[128] N. T. Dinh and Y. Kim, “Potential of information-centric wireless sensor and actor networking,” in *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, Jan 2013, pp. 163–168.

[129] A. Rayes, M. Morrow, and D. Lake, “Internet of Things implications on ICN,” in *2012 International Conference on Collaboration Technologies and Systems (CTS)*, May 2012, pp. 27–33.

[130] G. F. Marias, N. Fotiou, and G. C. Polyzos, “Efficient information lookup for the Internet of Things,” in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012, pp. 1–6.

[131] R. Ravindran, T. Biswas, X. Zhang, A. Chakraborti, and G. Wang, “Information-centric networking based homenet,” in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 1102–1108.



Literature overview

Table A.1: Sheet with all considered prior art in the literature study.

Title	Routing	Architecture	Caching strategy	Naming	Security	Mobility	Comparison	Misc	Survey
2017-2015									
[95]									
[96]	x			x			x		
[34]					x				
[97]								x	
[35]					x				
[38]					x				
[98]		x						x	
[24]		x						x	
[99]				x					
[100]				x				x	
[101]	x								
[28]		x							
[102]								x	x
[103]	x								x
[33]									x
[37]					x				
[32]			x	x					x
[104]					x	x			
[25]		x		x					x
[105]			x						
[36]		x			x				
[106]									x
[20]	x			x					
[21]	x								
[107]	x								
[108]								x	x
[26]	x	x							
[109]			x						
[29]		x						x	
[27]		x		x					
[110]								x	
[111]								x	
[112]								x	
[23]	x		x						
[113]		x							
[31]			x						
[114]							x		
[115]	x								
[116]					x			x	
2014									
[40]							x		
[117]							x		
[22]	x						x		x
[118]					x				
[119]									x
[120]				x		x			
[121]								x	
[122]		x			x	x			
[123]	x							x	
[124]	x							x	x
[125]								x	
[126]									x
<2014									
[127]								x	
[128]		x							
[129]									x
[130]		x							
[131]				x	x				x
Total number of papers	12	12	5	8	9	3	5	16	12

B

Implementation

B.1. BRITE configuration

```
BeginModel
        Name = 5
        edgeConn = 1
        k = -1
        BWInter = 1
        BWInterMin = 1024.0
        BWInterMax = 1024.0
        BWIntra = 1
        BWIntraMin = 1024.0
        BWIntraMax = 1024.0
EndModel

BeginModel
        Name =  4
        N = 20
        HS = 1000
        LS = 100
        NodePlacement = 1
        m = 2
        BWDist = 1
        BWMin = 1024.0
        BWMax = 1024.0
EndModel


BeginModel
        Name =  1
        N = 20
        HS = 1000
        LS = 100
        NodePlacement = 1
        GrowthType = 1
        m = 2
        alpha = 0.64
        beta = 0.47
        BWDist = 1
        BWMin = 1024.0
        BWMax = 1024.0
```

```
EndModel

BeginOutput
        BRITE = 1          #1/0=enable/disable output in BRITE format
        OTTER = 0          #1/0=enable/disable visualization in otter
EndOutput
```

C Results

C.1. ‘Smart home’

Hop count

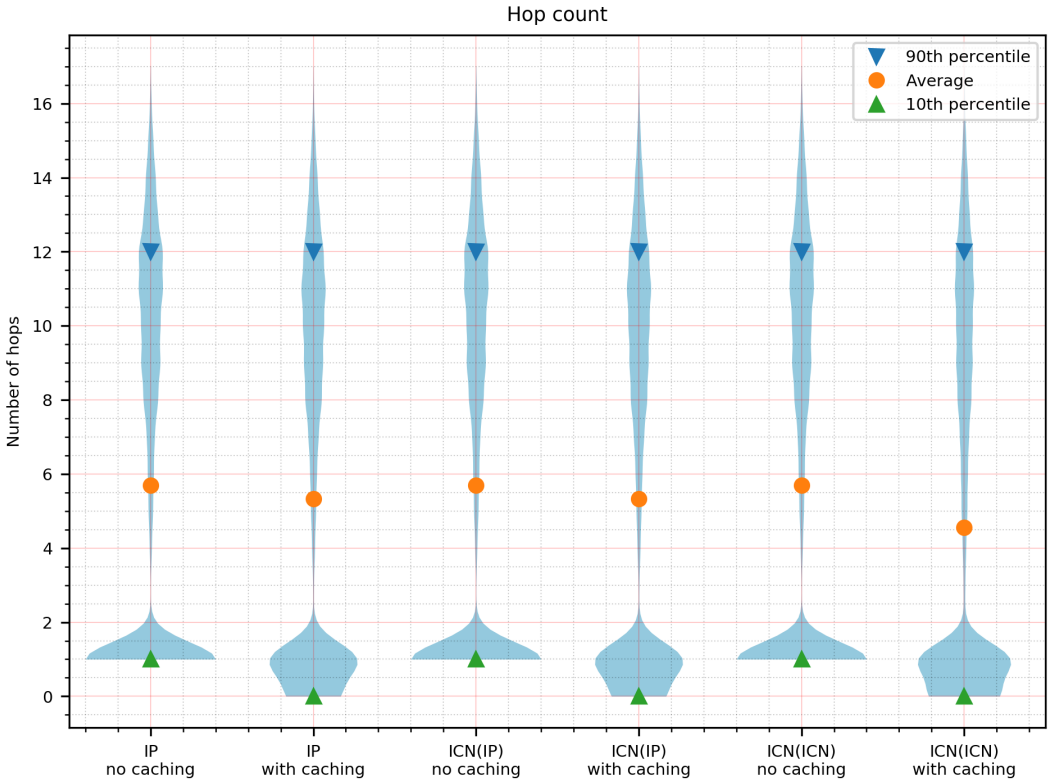


Figure C.1: Hop count results for the ‘smart home’ baseline scenario. The blue shape shows the kernel density estimate (KDE) for the delay distribution. The green error bars indicate the 95% confidence intervals for the hop count.

C.2. ‘Smart factory’

Hop count

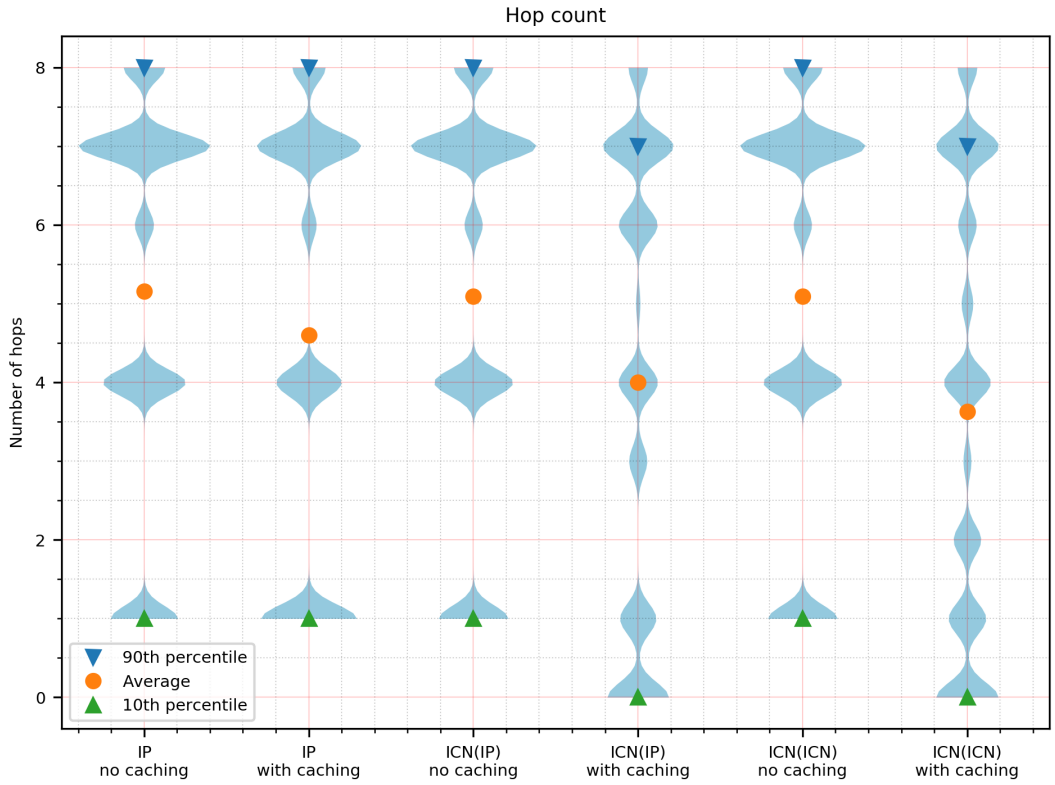


Figure C.2: Hop count results for the ‘smart factory’ baseline scenario. The blue shape shows the kernel density estimate (KDE) for the delay distribution. The green error bars indicate the 95% confidence intervals for the hop count.