On information-centric routing and forwarding in the internet of things

Marcel Enguehard

université PARIS-SACLAY

NNT : 2019SACLT013

Thèse de doctorat

TELECOM ParisTech

On Information-Centric routing and forwarding in the Internet of Things

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom ParisTech

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Réseaux, Information et Communications

Thèse présentée et soutenue à Paris, le 15 avril 2019, par

MARCEL ENGUEHARD

Composition du Jury :

Thomas Clausen Professeur, École Polytechnique	Président
George Pavlou Professeur, University College London	Rapporteur
Lan Wang Professeur, University of Memphis	Rapporteur
Emmanuel Baccelli Chercheur, INRIA	Examinateur
Matthias Wählisch Professeur, Freie Universität Berlin	Examinateur
Lixia Zhang Professeur, University of California, Los Angeles	Examinateur
Jean-Louis Rougier Professeur, Télécom ParisTech	Directeur de thèse
Giovanna Carofiglio Distinguished Engineer, Cisco Systems	Co-encadrante

On Information-Centric routing and forwarding in the Internet of Things

Marcel Enguehard

Mal nommer un objet
c'est ajouter au malheur
de ce monde

ALBERT CAMUS, 1944

Acknowledgements

First and foremost, I would like to express my utmost gratitude to Dr. Giovanna Carofiglio et Prof. Dario Rossi for supervising my thesis during the major part of the last three years. Dario's approach, mixing wide-ranging technical expertise, a thorough knowledge of academia, and an always positive mood has been of the main reasons I was able to conduct my PhD successfully. Giovanna's vision, rigour, and leadership made her essential to the success of my thesis. Her guidance was crucial in defining and focusing the research directions that I explored, but also to considerably increase the technical and writing quality of my scientific production, including this manuscript. She was an amazing mentor during my employment at Cisco, helping me naviguate the company and bootstrap my career. I would also like to warmly thank Ralph Droms, who supervised my during the six first months of my thesis. Our conversations put me on the right path and I learned a lot during them. I must also thank him for taking the time to finish the work we started together after he moved on to other occupations. Finally, I would like to thank Jean-Louis Rougier for guiding me through the last steps of my PhD. His advice was key in the preparation of my manuscript and defense.

I would like to extend my gratitude to my defense committee, and first to the reviewers Prof. George Pavlou and Prof. Lan Wang for taking the time to read and thoroughly comment my thesis manuscript. I would also like to thank the examiners Prof. Thomas Clausen, Dr. Emmanuel Baccelli, Prof. Matthias Wählisch and Prof. Lixia Zhang for participating in my PhD defense and for asking sharp and acute questions which led to very interesting discussions.

Il me faut maintenant remercier mes collègues et amis grâce à qui j'allais toujours au travail de bonne humeur. Merci d'abord à l'ensemble des «jeunes »du PIRL : Aloÿs, Guillaume, Jacques, Mohammed, Nathan, Pierre, Victor, et Yoann. Mentions spéciales à Yoann Desmouceaux, avec lequel nous avons beaucoup collaboré quitte à programmer dans des langages ésotériques et à Jacques Samain avec lequel nous avons partagé beaucoup de moments au sein de l'équipe ICN. J'en profite pour remercier aussi l'ensemble des membres de l'équipe ICN avec lesquels j'ai eu beaucoup de plaisir à travailler, tant pour leurs qualités humaines que pour leurs compétences professionnelles. Merci en particulier à Jordan Augé, pour sa motivation et son ardeur à toute épreuve et pour toutes ces conversations pendant lesquelles j'ai beaucoup appris. Merci enfin à l'ensemble des membres du PIRL, et en particulier Mark Townsley pour sa bienveillance et son intérêt depuis mon recrutement jusqu'à ma soutenance. Mark a rendu ma thèse possible et a toujours été présent lorsque des difficultés se sont présentées.

Merci aussi à l'ensemble de l'équipe de la chaire NewNet@Paris à Télécom,

qui rendirent les moments passés au laboratoire toujours plus agréables et amicales. Je dois évidemment beaucoup à l'ensemble de mes co-auteurs, dont beaucoup ont déjà été cités, pour leur contribution inestimable à la conception et rédaction des différentes publications présentées dans cette thèse : Jordan Augé, Giovanna Carofiglio, Alberto Compagno, Yoann Desmouceaux, Ralph Droms, Luca Muscariello, Victor Nguyen, Pierre Pfister, Dario Rossi, Mauro Sardara, Wenqin Shao, Éric Vyncke.

Je voudrais remercier mes proches qui m'ont soutenus tout au long de ma thèse : ma famille, et en particulier mes parents qui m'ont encouragé à m'inscrire en thèse, à mes amis qui ont pu se libérer pour assister à ma soutenance. Enfin et surtout, merci à Margot qui a été là durant toute ma thèse, pour fêter les moments heureux mais aussi pour me soutenir aux moments difficiles et qui m'a toujours écouté et su me conseiller avec pertinence quand je doutais.

Résumé / Abstract

Résumé en Français

Les réseaux centrés contenus (ICN) sont considérés comme une solution aux nouveaux défis et modes de communication liés à l'émergence de l'Internet des Objets (IoT). Pour confirmer cette hypothèse, la problématique fondamentale du routage sur les réseaux ICN-IoT doit être abordée. Cette thèse traite de ce sujet à travers l'architecture IoT.

Premièrement, une méthode sécurisée est introduite pour acheminer des paquets ICN à partir de coordonnées géographiques dans un réseau sans-fil de capteurs à faible puissance. Elle est comparée à une inondation optimisée du réseau inspirée des approches existant dans la littérature. En particulier, leur faisabilité et passage à l'échelle sont évalués via un modèle mathématique. Le modèle est paramétré grâce à des données réalistes issues de simulation, de la littérature, et d'expériences sur des capteurs. Il est montré que le routage géographique permet de diviser la mémoire nécessaire sur les capteurs par deux et de réduire considérablement le coût énergétique du routage, en particulier pour des topologies dynamiques.

Ensuite, ICN est utilisé pour contrôler l'admission à une plate-forme de calcul de type Fog afin de garantir le temps de réponse. La stratégie de contrôle d'admission proposée, le LRU-AC, utilise l'algorithme Least-Recently-Used (LRU) pour apprendre en direct la distribution de popularité des requêtes. Son efficacité est démontrée grâce à un modèle fondé sur un réseau de files d'attente. Une implémentation du LRU-AC est proposé, utilisant des filtres de Bloom pour satisfaire aux contraintes des cartes FPGA. Son bien-fondé est prouvé par un modèle mathématique et son efficacité en termes de latence et débit démontrée.

Enfin, on présente vICN, un outil pour la gestion et la virtualisation de réseaux ICN-IoT. Il s'agit d'une plate-forme qui unifie la configuration et la gestion des réseaux et des applications en exploitant les progrès des techniques d'isolation et de virtualisation. vICN est flexible, passe à l'échelle, et peut remplir différents buts : expériences à grande échelle reproductibles pour la recherche, démonstrations mélangeant machines émulées et physiques, et déploiements réels des technologies ICN dans les réseaux IP existants.

Abstract

As the Internet of Things (IoT) has brought upon new communication patterns and challenges, Information-Centric Networking (ICN) has been touted as a potential solution. To confirm that hypothesis, the fundamental issue of routing and forwarding in the ICN-IoT must be addressed. This thesis investigates this topic across the IoT architecture.

First, a scheme to securely forward ICN interests packets based on geographic coordinates is proposed for low-power wireless sensor networks (WSN). Its efficiency is compared to an optimized flooding-based scheme similar to current ICN-WSN approaches in terms of deployability and scalability using an analytical model. Realistic data for the model is derived from a mixture of simulation, literature study, and experiments on state-of-the-art sensor boards. Geographic forwarding is shown to halve the memory footprint of the ICN stack on reference deployments and to yield significant energy savings, especially for dynamic topologies.

Second, ICN is used to enhance admission control (AC) to fixed-capacity Edge-computing platforms to guarantee request-completion time for latency-constrained applications. The LRU-AC, a request-aware AC strategy based on online learning of the request popularity distribution through a Least-Recently-Used (LRU) filter, is proposed. Using a queueing model, the LRU-AC is shown to decrease the number of requests that must be offloaded to the Cloud. An implementation of the LRU-AC on FPGA hardware is then proposed, using Ageing Bloom Filters (ABF) to provide a compact memory representation. The validity of using ABFs for the LRU-AC is proven through analytical modelling. The implementation provides high throughput and low latency.

Finally, the management and virtualization of ICN-IoT networks are considered. vICN (virtualized ICN), a unified intent-based framework for network configuration and management that uses recent progress in resource isolation and virtualization techniques is introduced. It offers a single, flexible and scalable platform to serve different purposes, ranging from reproducible large-scale research experimentation to demonstrations with emulated and/or physical devices and network resources and to real deployments of ICN in existing IP networks.

Contents

List of Acronyms

ABF	Ageing Bloom Filter
AC	Admission Control
API	Application Programming Interface
CS	Content Store
CSP	Constraint Satisfaction Problem
DB	Data base
DAG	Directed Acyclic Graph
DMA	Direct Memory Access
DoS	Denial of Service
F&L	Flood and Learn
FIB	Forwarding Information Base
FSM	Finite-State Machine
IBN	Intent-based networking
ICN	Information-Centric Networking
IoT	Internet of Things
IRM	Independent Request Model
ISP	Internet Service Provider
ITS	Intelligent Transportation Systems
LAN	Local-Area Network
LPM	Longest-Prefix Match

LFU	Least-Frequently Used
LRU	Least-Recently Used
M2M	Machine-to-Machine
MAC	Message Authentication Code
MANO	Management and Orchestration
MPR	Multi-point relay
MTU	Maximum transmission unit
NAT	Network Address Translation
NDN	Named-Data Networking
NFN	Named-Function Networking
NFV	Network Function Virtualization
OS	Operating system
PIT	Pending Interest Table
QoS	Quality of Service
RTT	Round-Trip Time
SDN	Software-Defined Networking
SLA	Service-level agreement
TLV	Type-Length Value
V2V	Vehicle-to-vehicle
VM	Virtual-Machine
VNF	Virtual Network Function
WSN	Wireless Sensor Network

Chapter 1

Introduction

The underlying principles of the Internet have remained rather stable since its inception. TCP and IP, introduced in 1974 by Cerf and Kahn [1], still remain the fundamental protocols upon which communication channels are built. The use and the scale of the Internet, however, has dramatically evolved, moving from kilobytes to petabytes per second, from dozens to billions of connected end-points. Nowadays, networks have in fact invaded most aspects of our society and everyday life. There are but a few examples more representative of that phenomenon than the emergence of the Internet-of-Things (IoT), which promises to help automate and optimize our environment by increasing the number of connected devices by another few orders of magnitude.

The scale of the envisioned IoT deployments and their requirements in terms of bandwidth, latency, flexibility have raised concerns about the suitability of TCP/IP. New architectures have thus been proposed to replace or enhance TCP/IP so as to accommodate these new requirements, chief among them Information-Centric Networking (ICN). This thesis, in line with this approach, investigates the suitability of ICN for the IoT from a networking perspective.

1.1 The Internet-of-Things

The past ten years have seen the emergence of the so-called "*Internet-of-Things*" (IoT). The turn-of-phrase, which originally describes the growing trend of providing Internet connectivity to everyday objects (e.g., kitchen appliances, transportation tools, medical material), has been used to describe many different applications. The connectivity is used to build a network of "smart" objects that cooperate in view of automating specific tasks and systems. Notable examples of IoT subdomains are, among others, building [2,3], home [4–6], or factory [7,8] automation, intelligent transportation systems (ITS) [9], environmental monitoring [10], smart cities [11–13], the "smart grid" [14], etc. This section aims at gaining a principled understanding of the IoT despite that diversity, looking at IoT applications (Section 1.1.1), networks (Section 1.1.2), and finally at the challenges associated with both (Section 1.1.3).

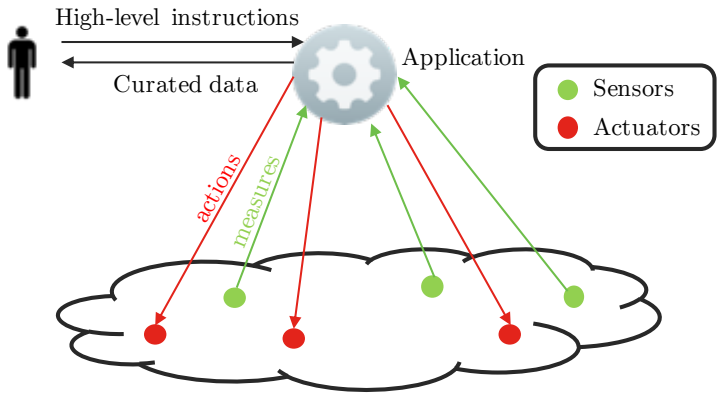


Figure 1.1 – Typical functioning of an IoT deployment

1.1.1 IoT applications

While diverse in form, in scale, and in objectives, IoT applications share common principles. Their goal is, in general, to automate and optimize part of our environment (the home, the city, etc.) As such, they follow the same global patterns of reading data from the environment, processing it to compute a metric, compare that metric against an objective, and apply actions on the environment accordingly.

These common characteristics lead IoT applications to share the same fundamental building blocks. Indeed, they are composed of four main actors: sensors, actuators, applications, and humans. Their respective function, summarized in Figure 1.1 can be described as such:

Humans deploy and manage IoT deployments. They set high-level goals (e.g., keep the building temperature at 20 °C) for automation or expect curated data (e.g., time series of the average level of pollution in a city) from the deployment.

Sensors are the basic data generators in IoT networks. They are typically single-purpose hardware modules deployed in a system, tuned to measure a specific physical phenomenon (e.g., temperature, speed, humidity). For instance, current top-of-the-line cars embed hundreds of sensors [15].

Actuators are modules that perform actions to influence the state of the system, for instance, by controlling the heating system in a specific room or the braking system of a car. They are controlled using the data generated by the sensors as a feedback mechanism.

Applications are used to provide intelligence to the system. They take as input the data measured by the sensors and the human-issued high-level objective and output instructions for the actuators, curated data for the humans, etc.

1.1.2 IoT networks

Another common thread of the various IoT use cases is their network architecture. Indeed, an IoT deployment usually implicates various logical and

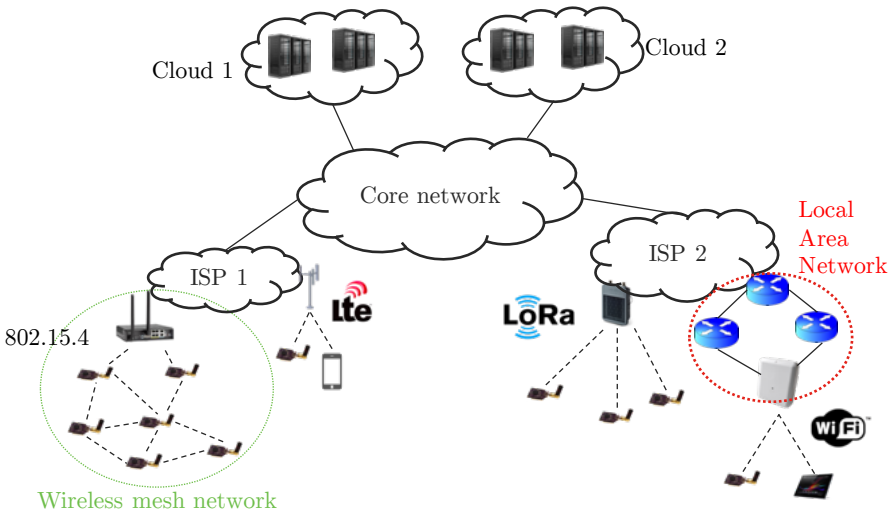


Figure 1.2 – Examples of IoT deployments

topological locations of the Internet. In particular, four main components can be identified: the IoT network, the access network, the core network, and the data center. The articulation between these components is shown in Figure 1.2

The *IoT network* is where sensors and actuators are deployed. It corresponds to the geographic location of the system controlled or measured by the IoT (e.g., smart building, city, etc.). Sensors and actuators are often low-power nodes, with constrained computing, memory, and energy resources. They are equipped with a network interface, through either wired (Ethernet, Power Line Communications) or wireless (IEEE 802.11, IEEE 802.15.4, LoRa, cellular connections) technologies. IoT networks broadly follow one of two topology classes: star-shaped networks, and mesh networks. In star-shaped networks, all nodes are only connected to a *base station*, which acts as a relay towards both the inside and the outside of the IoT network. Typically, cellular or IEEE 802.11 networks are star-shaped. In mesh networks, on the other hand, nodes that are in communication range can communicate directly with each other. They form a decentralized network, where information can be transmitted over multiple hops from one sensor to another without going through a centralized base station. Such mesh networks of sensors, usually called Wireless Sensor Networks (WSN), have been widely studied in the literature [16].

To link the IoT network with the rest of the Internet, IoT operators usually buy connectivity from an *Internet Service Provider* (ISP), connecting either the devices directly to the ISP network or via their own Local-Area network (LAN). The ISP network holds a specific place compared to other networks traversed by the IoT traffic: it is the only network operator with which the IoT operator maintains a business relation.

While sensors and actuators are usually only found in the IoT network, applications are ubiquitous in the IoT context. A popular solution is, for instance, to rent compute and storage resources in a *Cloud platform*. In this case, operators deploy their applications in a virtualized environment, typically hosted in a large data-centers remote from the actual IoT network. Such a solution is often

Table 1.1 – Location of IoT building blocks

Device type	**Location**
Sensor	IoT network
Actuator	IoT network
Applications	Ubiquitous
Human-facing devices	IoT network; Data center network

practical for, e.g., collecting long time-series over specific readings or performing hard computations over large sets of data. On the other end of the spectrum, some rudimentary logic can be deployed directly on a sensor or an actuator. In this case, actuators can, for instance, retrieve data directly from sensor boards using horizontal Machine-to-Machine (M2M) communications. This approach is fit for low-latency control loops that require only minimal (if any) data processing. Finally, as a class of compute-intensive low-latency IoT applications started to emerge, e.g., for Augmented/Virtual reality (AR/VR) or for ITS, a new computing platform has emerged: *the Fog* [17]. The Fog is defined as a highly virtualized, often distributed, computing and storage layer that sits at the edge of the network, e.g., in the IoT operator LAN or in the ISP network. Fog deployments are thus ideal for applications that require a low response time or, for instance, geographic awareness.

Finally, ISP and Cloud are connected through the Internet core network, sometimes through multiple transit networks. Obviously, and as represented in Figure 1.2, real-life IoT deployments can consist of multiple IoT networks connected to multiple access networks, with applications deployed in multiple data-center networks. In particular, the location of the actors in an IoT deployment is summarized in Table 1.1.

1.1.3 Challenges

The IoT has raised new challenges for communications networks. This section details some of the most important ones, namely new communication patterns, implementability, mobility handling, scalability, security, and Quality of Service (QoS). For each of these challenges, limitations of IP-based approaches are also presented.

Communication patterns

IoT traffic follows novel patterns, for which current networks have not been designed. A first novelty is the rise of *horizontal M2M traffic* [18], as opposed to the traditional vertical client-server interactions seen in today's Internet. Indeed, while current Internet connections mainly consist of human-facing terminals (e.g., Web browsers or video players) in edge networks downloading data from data-centers in the cloud, the rise of the IoT has seen autonomous devices sharing information with each other. M2M communications can include, for instance, sensors sharing their data with an actuator or cars sharing traffic information with each other. In fact, between 2016 and 2021, Cisco expects 51 per cent of worldwide connected devices to perform M2M communications [19].

Furthermore, IoT streams are quite different from the original large point-to-point file transfers for which, e.g., TCP was designed. They often consist in periodic transfers of low-size data blobs in a one-to-many fashion, where multiple applications subscribe to consecutive readings of a sensor [20, 21]. To scale such communications patterns, *multicast* support is essential [22]. Native network multicast is even more important in constrained networks with multiple devices competing for network access and where thus network utilization should be minimized. In IP, multicast is not natively supported and comes with a large signalling overhead and strong limitations [23].

Implementability

Classical IP stacks are notoriously hard to scale on low-power IoT nodes with constrained capabilities (in terms of battery, compute power, storage capacity, network bandwidth). Not only is the IP stack memory-consuming [24], but its headers are disproportionally large for the low maximum transmission unit (MTU) of low-power networking technologies [25]. Thus, a set of adapted protocols has been standardized for low-power networks: 6LoWPAN [25] for the network layer with RPL [26] as a routing protocol, DTLS [27] as secure transport, CoAP [28] for the application layer. These protocols were designed with IP compatibility in mind but are still ill-fitted to the IoT context. For instance, Clausen et al., have argued that RPL is sub-optimal for M2M communications [29]. Furthermore, low-power nodes connected over wireless technologies are often disconnected, be it because they sleep to save battery or because of interference over the wireless medium. This is particularly problematic for the connection-based DTLS transport, which cannot handle such connectivity interruptions [23].

Mobility

Mobility is an integral part of many IoT use cases. The smart city or ITS, for instance, involve nodes with high physical mobility. Even in static deployments, low-power nodes connected via wireless media only have intermittent connectivity due to their low duty-cycle or to wireless interference. Handling this mobility is a notably hard challenge in IP, as IP addressing is location-based [30].

Scalability

As explained in Section 1.1.1, a key feature the IoT is the deployment of many sensors and actuators in our everyday environment and the introduction of Internet connectivity to previously monolithic appliances. The number of devices connected to the Internet is thus expected to skyrocket, reaching 27.1 billion in 2021 [19]. These devices will generate and consume data and raise scalability issues in multiple aspects of the network. Beyond the obvious problem of being able to efficiently fit all of this data into the current networking infrastructure, it also brings traditional network protocols under tremendous stress. Routing protocols, for instance, could encounter scalability issues, e.g., from routing tables exploding in size or from control traffic overwhelming the network.

Table 1.2 – Classification of IoT applications by QoS requirements

Application class	**QoS requirement**	**Example use case**
Latency-critical	1-10 ms	Factory automation
Latency-sensitive	100-500 ms	Building automation
Latency-tolerant	None	Environmental monitoring

Security

Security is paramount for IoT applications, which present a particularly wide attack surface [31]. The connection-based security model built on (D)TLS is problematic for IoT environment, as creating and maintaining the channel requires important resources (in terms of network, memory, and compute) on constrained devices [32]. It is also hard to accommodate alongside network multicast and mobility, which as explained earlier are essential features for the IoT. Furthermore, as IoT verticals are often built using, e.g., more powerful relay nodes such as gateways as protocol proxies [31, 33], a purely connection-based approach does not guarantee end-to-end authentication and data integrity between the sensor and the data consumer.

Quality of Service

Finally, the IoT is characterized by the cohabitation of many different QoS requirements. IoT applications can roughly be categorized into three classes, as summarized in Table 1.2: (i) *latency-critical*, when data must be received within 1-10 ms (typically in M2M interactions [21]), (ii) *latency-sensitive*, where the timescale interaction is in the order of 100 ms (e.g., applications with human interaction [34]), and (iii) *latency-tolerant*, that have no specific delay constraint such as time-series collection for big-data analysis. All of these applications coexist along the IoT vertical and sometimes even rely on the same underlying sensor data.

1.2 Information-Centric Networking for the IoT: motivation

To tackle the challenges listed in Section 1.1.3, recent research has put forward a new network architecture called Information-Centric Networking (ICN) [23, 24, 35, 36]. Section 1.2.1 presents the high-level functioning of ICN. Theoretical advantages of ICN for the IoT are then presented in Section 1.2.2.

1.2.1 Information-Centric Networking

ICN is a recent network paradigm introduced by Van Jacobson et al. [37]. It is centred on the idea of using *identifiers* rather than *locators* as the central network primitive. Indeed, while Pouzin, inventor of the first datagram network, already pointed out in 1973 that topological addresses had limitations [38], current IP networks are still based on host-to-host communications. Conversely, in ICN, the central unifying layer is *named data*. The network is then in charge

of locating and retrieving the data associated with a given name rather than forwarding a packet to a given destination address.

Indeed, each addressable object in the network is given a name (e.g., */Cisco/ILM/Eiffel/Temp* for the temperature in the Eiffel room of the Cisco office in Issy-les-Moulineaux). Names are rather generic and can represent anything: a specific sensor reading, a chunk of a video, a remote procedure call on an actuator, or even an endpoint. In ICN, network addresses thus represent named objects rather than only the host where said objects can be found. The forwarding plane is then in charge of finding *a* location for the named object. While ICN can be found in many derivations [39–42], in this thesis the term ICN refers to two rather similar realisations of ICN: CCNx [43, 44] and Named-Data Networking (NDN) [40]. Both architectures use hierarchical naming conventions (similarly to DNS) and are based on a pull-based communication model with symmetric routing. They rely on two fundamental packet types: *Interest* and *Data* packets. Interest packets, which are generated by *consumers*, contain the name of the piece of content that the consumer wishes to access. It is forwarded over the network based on that name towards the *producer* of the data until it reaches a point where the corresponding content is available. There, the content is put into a Data packet, also addressed by the content name, which is forwarded back to the consumer using the reverse path of the Interest packet.

ICN routers are based on three data structures: the Forwarding Information Base (FIB), the Pending Interest Table (PIT), and the Content Store (CS). The FIB, similarly to an IP router, maps name prefixes to next hop in the ICN network. It is used to forward Interests towards their destination object, by matching the name in the Interest against the name prefixes in the FIB using Longest-Prefix Match (LPM). The use of hierarchical names and LPM helps the FIB scale, by, for instance, aggregating all the routes towards the Cisco Issy-les-Moulineaux office under the */Cisco/ILM* name prefix. The PIT is used to record in-flight Interest to perform symmetric routing of the corresponding Data packet. It contains a list of all the names for which the router has received an Interest packet that has not been satisfied yet with a Data packet. Each of the names is stored with the incoming and outgoing interfaces of the corresponding Interest packets. These Interests are grouped by name: all requests for the same named object are *aggregated* in a single PIT entry. Namely, if a router receives an Interest for a name that is already in the PIT, it does not forward the Interest, but only update the PIT entry to register the current Interest incoming face. Interest aggregation is a fundamental feature of ICN, as the name-based forwarding allows to detect requests for the same name object and achieve *native multicast*. Upon reception of a Data packet, its name is matched in the PIT against outstanding Interests. The Data packet is then forwarded to all the registered incoming interfaces, thus taking the reverse path of the corresponding Interests. Finally, the CS is used to cache content directly on the forwarding path. Upon receiving an Interest, the router checks whether the requested content is available in its CS and serves it directly from its cache in that case. Upon receiving a Data packet, a cache admission policy (e.g., Least-Recently-Used (LRU) eviction) is used to decide whether to store the corresponding piece of content in the CS or not. The forwarding mechanism for Interest packets is summarized in Figure 1.3.

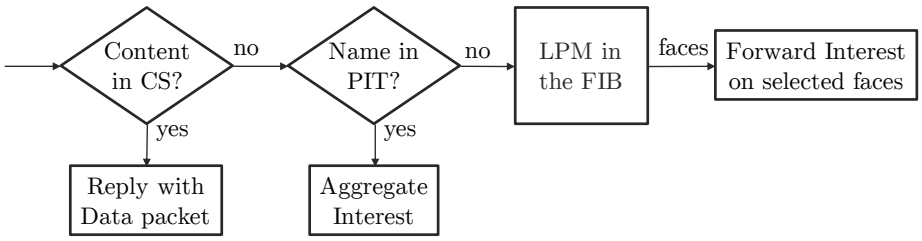


Figure 1.3 – Flowchart of ICN Interest forwarding

1.2.2 ICN for the IoT

IoT has been identified as an area where ICN could bring many advantages [45]. Among others, name-based forwarding, in-network caching and multicast, object-based security, and mobility support are characteristics of ICN that could be beneficial for the IoT.

Name-based forwarding

First of all, forwarding on data-identifiers rather than network locators seems suited to IoT networks. Indeed, enforcing names at the network layer forces applications to share a *common semantic*. This is highly desirable for sensor data to be shared between multiple applications, especially on single-purpose low-power sensors [21, 28, 36]. Inversely, network names allow applications to benefit from *in-network service discovery* [46]. This is extremely important in the IoT context, where applications are not necessarily aware of sensors deployed in the field [47]. It means that multiple sensors can provide the same data, the same service, and that content ubiquity is handled directly at the network layer [36]. This approach is more integrated than IP-based approaches like CoAP, which require layers of indirection to map services to IP addresses [28].

In-network caching and multicast

Furthermore, content ubiquity means that *caching* becomes a native feature in ICN networks [48]. Caching is a crucial feature for IoT applications. First, it helps increase data availability in WSN. Indeed, on-field low-power sensors might only have intermittent reachability (because, e.g., of low duty-cycles and unreliable wireless connectivity) [49]. Offloading the data produced by these sensors to more powerful and reliable nodes is necessary to ensure continuous data availability. Furthermore, caching is necessary to scale ever-growing IoT deployments. For instance, in WSNs, in-network caching allows for limiting the number of network transmissions and thus saving energy [50]. More generally, caches bring data close to its consumer, thus decreasing both download time and network utilisation. This is particularly crucial for low-latency applications (e.g., Augmented/Virtual Reality) [17, 36]. The efficiency of the data retrieval process is further increased thanks to the native multicast provided by the PIT and CS as described in Section 1.2.1.

Object-based security

This approach is furthermore enabled by the *object-based security* model of ICN, which maintains end-to-end trust between the data producer and the data consumer, even if the data goes through relay nodes [35]. Indeed, in ICN, security is associated with the data rather than with the channel. In practice, it means that every piece of data is signed and/or encrypted directly by its producer. The security context is thus attached to the object rather than to the host (as it is the case in TLS). This removes the limitations of TLS-based approaches described in Section 1.1.3. Furthermore, ICN security scales with the number of produced objects rather than with the number of connections handled by a machine [32,36]. Low-power sensor nodes are freed from the burden of maintaining high-cost TLS connections, thus lowering the energy overhead of securing the data.

Mobility support

Network mobility is a core feature of IoT networks, especially for use cases as ITS or Smart Cities. In IoT networks, mobility is not only associated to moving devices (e.g., connected vehicles or pedestrian-held devices), but also network mobility due to unreliable physical connections (e.g., wireless interferences on an IEEE 802.15.4 channel). The connection-less model of ICN is an obvious candidate to handle that mobility [51]. Indeed, consumer mobility is naturally handled thanks to symmetric routing: since consumers do not have network addresses, no signalling is required in case of a mobility event. More generally, the simplicity of the ICN stack and its native collaboration with the way applications are designed is expected to reduce signalling and decrease the number of indirection layers [35].

1.3 ICN for the IoT: background

Research on ICN for IoT has focused on many different topics. This section summarizes seminal research contributions in three main areas that are of relevance for this thesis: ICN for the WSN (Section 1.3.1), ICN for the Fog (Section 1.3.2), and ICN for specific IoT applications (Section 1.3.3). For a more general survey on the ICN-IoT literature, the interested reader is referred to Arshad et al. [51].

1.3.1 ICN for the WSN

Data-centric approaches have a long history in WSN [52]. Of particular note in this context, Directed Diffusion is a data-centric routing protocol where all nodes perform application-aware routing [53]. In particular, sensors are identified by attribute-value pairs and, similarly to ICN, data retrieval is done through a pull-based model using Interest messages. Interests are either flooded, directed based on geographic location, or forwarded based on routes learned during previous Interest/Data exchanges. Directed Diffusion has been adapted to the ICN context by Amadeo et al. [54]. They propose an extended naming scheme that consists of a multi-dimensional attribute-value set (e.g., task type, location, task time). A hybrid approach between multi-dimensional sets and

traditional hierarchical names for IoT networks is proposed by Ascigil et al. [55]. Namely, a one-dimensional name is used as a network prefix in combination with a named-function (i.e., some processing to be applied on the retrieved data) and tags to describe the retrieved data. The prefix is used to route the packet towards a specific network location, where the tags are understood in context to find relevant sensors. The function is then used to enable in-network data processing. Similarly, Abidy et al., propose to extend the ICN naming scheme to perform in-network aggregation of several pieces of content [56].

Other authors have looked at adapting in a more straightforward way ICN-implementations to WSNs. Baccelli et al. [24], consider the performance of an out-of-the-box ICN stack (CCN-Lite [57]) over constrained nodes using the open-source operating system (OS) RIOT-OS [58]. Using an experimental approach, they show advantages of ICN in terms of implementation complexity (lower memory usage, lower header footprint in the packet) with respect to 6LoWPAN. They propose two naive forwarding strategies for ICN: vanilla flooding and "*flood-and-learn*", which consists in flooding Interest packets for which there is no FIB entry and learning routes by observing the hop from which the corresponding Data packet comes back. Similarly, Gündogan et al. [59], compare ICN with current IP-based standards for WSN (CoAP [28] and MQTT [60]). They show that ICN yields benefits in terms of implementability (i.e., footprint on the low-power device) and highlight an interesting trade-off: while ICN is more robust and resilient in multi-hop mesh scenarios, IP-based protocols are quicker and have less overhead in star-shaped deployments. Inversely, in [61], an implementation of COAP over ICN is proposed for enabling application interoperability in ICN-IoT networks. The authors show, in particular, that the native ICN-multicast allows for significantly decreasing the communication overhead. Shang et al., propose an alternative implementation of the NDN protocol suite over RIOT-OS [62], complemented by a larger framework to apply ICN for WSN [63]. Their architecture includes, among other contributions, a bootstrap protocol to issue meaningful names to sensors connecting to the IoT networks, a publish-subscribe system that respects the pull-model of ICN, and trust model for the data generated by the sensors. Looking specifically at constrained network technologies, Ren et al., present a lightweight CCNx-based protocol for fitting CCNx headers in 802.15.4 MTUs [64]. As the lower memory footprint of ICN stacks opens the opportunity for caching in IoT boards, Hahm et al. [49], study how cooperative caching can increase content-availability for nodes with low duty-cycles. Pfender et al., also look at caching for low-power nodes, focusing on the efficiency of admission and eviction policies for IoT data [65]. To further optimize the caches, Melvix et al., design a forwarding strategy where ICN forwarders map Interest names to *scopes* and *domains*, which describe the expected accuracy on the requested data [66]. For data that is tolerant to inaccuracies, stale or approximating pieces of content available in the CS can be used instead of forwarding the Interest in order to conserve energy.

Several pieces of work have also addressed the issue of security for ICN-WSN. In [67], the authors propose a global security framework for ICN-WSN that enables both push- and pull-interactions. The framework enforces access control over the ICN namespace and protects against Denial-of-Service (DoS) attacks using either asymmetric or symmetric cryptography. Compagno et al., propose OnboardICNg, a mechanism for secure-network join tailored for low-power IoT devices [68]. The evaluation shows improvements in terms of energy

consumption and onboarding speed compared to IP-based standards. Similarly, Mick et al., introduce LaSeR, a lightweight scheme for sensor onboarding in the context of Smart Cities [69]. In particular, LaSeR is integrated into the routing protocol so as to provide both functionalities (routing and authentication) at a lower overhead.

1.3.2 ICN for the Fog

The first natural use case for the Fog in an ICN network is as a caching layer. In [48], the authors propose a metric for optimizing content placement in distributed and collaborating Fog caches. Their metric is based on content centrality, i.e., how well connected a cache is to the content it is serving. Hit-rate improvements are shown compared to topology-based (rather than content-based) centrality metrics and to non-collaborative caching systems. Their scheme is extended in [70] by considering popularity in the centrality metric. Wang et al. [71], propose to use the Fog as a classifying layer to help routers decide which data to cache. They use the Fog as a computing layer that does not suffer from the same limitations as ICN routers to distinguish dynamic data (which should not be cached) from static data (which should be cached). That classification is then encoded as a tag in the Data packet header and used by routers to optimize the content of their CS.

Other authors have looked at how to exploit the Fog to realise one of the promises of ICN: in-network processing. One of the main steps in that direction is Named-Function Networking (NFN), an extension of ICN where packets do not only carry the name of the desired content but also the name of a function to be applied to said content [72]. Simple data-processing (e.g., *lambda functions* [73]) can then be fetched from a function repository to perform opportunistic in-network processing of data. The named functions can, for instance, be implemented as Virtual Machines (VM) [74]. ICN routers equipped with a hypervisor can then launch these VMs on the content stored in a data packet to perform some computation on it. Scherb et al., specifically propose to apply NFN in the IoT by prioritizing Fog nodes to execute functions as on-path routers have only limited computing capacities [75].

Finally, some authors have explored the interaction between ICN and Fog from a network point-of-view. Shang et al., argue that the ICN naming and security model offers the opportunity to break from Cloud-only approaches by deploying applications directly into edge networks [76]. The Cloud is used as a complementary platform that can be used if available but is not necessary for the IoT applications to run, thus protecting the latters from loss of connectivity. Adhatarao et al., suggest using a Fog gateway as a protocol translation node between standard ICN protocols and a lightweight counterpart used to fit the requirements of low-power IoT networks [77]. In [78], the Fog is used to perform content-aware filtering for security services in social networks. Specifically, the Fog layer acts as a security middleware between two parties of an ICN-enabled social network to discard illegal content by classifying the exchanged content along labels defined by a regulator. Nguyen et al. [79] suggest ICN as a natural network layer to enable horizontal Fog-to-Fog communications. They use two case studies (video streaming and building energy management) to show the theoretical advantages of ICN in the Fog-to-Fog context such as reduced signalling, latency, and network utilization.

1.3.3 ICN for specific IoT applications

Finally, ICN has been studied as an enabler for specific applications: smart homes and buildings, smart cities, and ITS. This section contains an overview of the literature on applying ICN to these use cases. Note that, while substantial work has also been conducted on smart healthcare [80–82], it mostly addresses application-layer naming and security, which are not the topic of this thesis. Thus, we do not cover it in this state-of-the-art.

In [83], an implementation of ICN for smart homes is presented. The authors suggest a flexible naming scheme and propose to support push operations by embedding data into Interest packets called *Interest notifications*. Similarly, Silva et al., evaluate the efficiency of Interest notifications for smart home lighting, showing a reduction in message-delivery delays compared with an HTTP-based platform [84]. Shang et al., investigate building automation and management systems [85]. They propose a namespace to represent both sensor data (based on its location in the building, its type, and a timestamp) and users (based on user-ids and security material). That naming scheme is used to perform access control by mapping the names of the users to the data namespaces that they are allowed to access.

Moving up in scale from smart homes, Piro et al., introduce a platform for ICN-enabled smart cities [86]. They propose a service discovery protocol that allows the user to retrieve the security context associated with a specific name and show its application to several smart city use cases. In [87], Yue et al., introduce DataClouds, a community-based ICN for smart cities. Communities are network overlays comprised of users interested in the same pieces of data. Splitting the ICN network into communities helps to scale the network by disseminating data and routes only to the interested users.

Another crucial part of the Smart City paradigm is the emergence of ITS. There, while ICN has again been identified as a natural solution for the network layer, several challenges are left to overcome [88, 89]. One particular area of interest in that regard is vehicle-to-vehicle communication (V2V). Yan et al., introduce an architecture that supports typical ITS communications patterns, which involve mobility, push-based communications, and large-scale retrieval of small-sized pieces of data [90]. Their architecture is based on a hierarchical topology, where nodes in the access network are responsible for specific geographic zones. These nodes are then used aggregate Data packets and segregate Interest packets in order to increase the scalability of the network. Their naming strategy is also based on using geographic zones as prefixes. Interest packets are then geographically routed towards the node in the access network that is in charge of the corresponding zone. Similarly, Grassi et al., propose a routing scheme for V2V communications based on geography [91]. Data is identified by a grid location in the Military Grid Reference System [92]. ICN forwarders map these grid locations to *geographic* faces using flood-and-learn techniques.

1.4 Thesis contribution

This thesis addresses the problem of routing and forwarding along the IoT vertical. This section summarizes the main contributions. First, the challenges tackled by the thesis are described in Section 1.4.1. Then, the contributions and

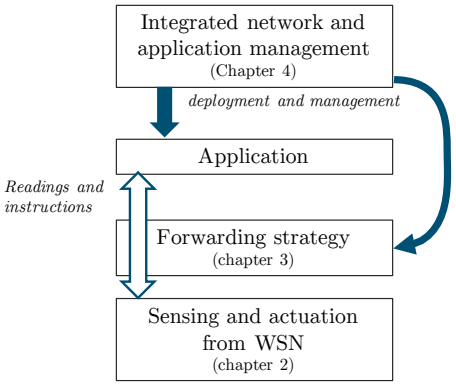


Figure 1.4 – Articulation between the contributions presented in the thesis

organisation of the thesis are summarized in Section 1.4.2.

1.4.1 Forwarding and routing in the ICN-IoT: challenges

Most of the issues described in Section 1.1.3 resonate in the way forwarding and routing are considered in the IoT: routing protocols must be scalable and responsive enough to manage mobility, forwarding must accommodate the various QoS requirements of IoT applications, the impact of both routing and forwarding on low-power IoT nodes must be considered, etc. On top of that, the use of ICN as the network layer for the IoT brings not only opportunities but new challenges. Two main problems arise: (i) the scalability of the ICN naming scheme in terms of routing structures (FIB) [93] and (ii) the development of forwarding strategies to select the next-hop when content is available in multiple locations [94]. The latter is of such importance to ICN that it has led to the development of a new block in the ICN router architecture: the strategy layer, used to transform the list of faces matched in the FIB into an actual forwarding decision [95]. As QoS is a predominant problem for the IoT, being able to coordinate forwarding strategies not only across nodes but also with application orchestration is of utmost importance [96].

While often mentioned as an important challenge [23, 24, 63], routing and forwarding for the IoT received little interest in the ICN community. For instance, most of the envisioned routing schemes for WSNs rely on a variant on a flood and learn mechanism, varying from simple name discovery [24, 86] to discovering groups prefixes based on attributes such as geographic location [54, 91] or tags [55]. Similarly, the literature on the ICN-enabled Fog has focused on exploiting the Fog resources to enhance ICN intrinsic capacities (e.g., larger storage space for caching or computing power for protocol translation) but has not considered how ICN routing and forwarding can be used to enhance the functionalities for which the Fog was built in the first place.

1.4.2 Contribution and organization

This thesis is divided into three main chapters, reflecting the respective contributions. In particular, routing and forwarding are considered across the

complete lifecycle of IoT data: data retrieval and processing. First, efficient data retrieval from IoT networks is investigated in Chapter 2. In particular, we explore the opportunities for performing geographic routing in the ICN-enabled WSN while keeping the network safe from malevolent attackers. In Chapter 3, ICN is used to bring intelligence to the network and help data processing platforms meet QoS requirements. In particular, we consider the problem of admission control in Fog platforms to enforce service time constraints. Finally, we address the broader problem of unifying network configuration and application orchestration in Chapter 4 by introducing vICN, a framework for ICN network management and virtualization based on recent progress in intent-based networking (IBN). The main contributions of this thesis are then summarized in Chapter 5, where perspectives on further work and research are briefly outlined. The articulation between the different contributions of this thesis is summarized in Figure 1.4.

Secure geographic routing for WSN

In a first step, we investigate the feasibility of using geographic forwarding to solve the routing problem in ICN-based WSNs. First, an architecture to securely perform geographic forwarding in the ICN semantic is introduced, based on four pillars: a naming scheme, a neighbour authentication mechanism, an ICN-based secure beaconing protocol, and a geographic forwarding algorithm. For the authentication mechanism, two protocols with similar security guarantees are considered: OnboardICNg [68], based on symmetric cryptography, and a new protocol, based on asymmetric cryptography. They are evaluated in terms of memory footprint and completion time. Second, an analytical model is presented to compare the efficiency of geographic forwarding to flood-and-learn approaches in terms of energy consumption, memory footprint, and CPU complexity. This model is completed by a mixture of experiments on standard sensor boards and of simulation to derive general guidelines as to when to use geographic forwarding with respect to flood-and-learn. In particular, it shows that geographic forwarding outperforms flooding-based approaches in terms of memory footprint in most cases and yields significant improvements in terms of energy overhead for dynamic topologies.

Popularity-based forwarding for Fog-Cloud placement

In a second step (and moving up the IoT vertical), we explore how ICN can help Fog platforms enforce service-level agreements (SLA). In particular, we look at how ICN can enhance Fog admission control (AC) by bringing application-awareness into the AC module. The Fog admission control problem is formalized as a constrained optimization problem using a queueing model. The LRU-AC, an approach based on estimating popularity through a virtual LRU cache, is proposed. Using the constrained optimization framework, the LRU-AC is shown to allow Fog nodes to accept more requests than application-blind approaches while respecting the same latency SLA. An implementation of the LRU-AC on programmable hardware is studied to provide AC at line-rate with minimal latency. This implementation is based on replacing the LRU structure with an Ageing Bloom Filter. The algorithmic validity of the implementation is proven through analytical modelling and verified through simulation. It is shown

to perform the LRU-AC with minimal latency overhead (3 µs) and line-rate throughput (16.7 Mpps).

Unified network management and application orchestration for the ICN-enabled IoT

In the last chapter, the challenge of unifying network and application management is explored. This problem is particularly stringent in the ICN-enabled IoT for many reasons, e.g., (i) the scale and diversity of IoT networks requires low-granularity control but also high-level human facing-interfaces; (ii) current ICN implementations and toolchains typically lack the comprehensive control mechanisms and protocols developed for IP; (iii) in ICN, the potential for interactions between the network and applicative layers reinforce the need for unifying network management and application orchestration. We thus introduce virtualized ICN (vICN), an intent-based framework for network virtualization and management developed specifically for ICN. It offers a single, flexible and scalable platform to serve different purposes, ranging from reproducible large-scale research experimentation to demonstrations with emulated and/or physical devices and network resources and to real deployments of ICN in existing IP networks. We highlight the flexible, modular, and scalable design of vICN and provide concrete examples to demonstrate its potential. Finally, the possibility of using ICN as a control protocol is explored. In particular, we propose an intent-based alternative to current IP-based network control standards that is able to perform routing and forwarding over orchestration objects. We illustrate benefits in terms of network programmability, reliability, and scalability.

1.5 Publications

Journal papers

— [97] M. Enguehard, Y. Desmouceaux, G. Carofiglio. "Efficient latency control in Fog deployments via hardware-accelerated popularity estimation". Under review in *ACM Transactions on Internet Technology*. 2019

— [98] M. Enguehard, D. Rossi, R. Droms. "On the cost of geographic forwarding for information-centric things". In *IEEE Transactions on Green Communications and Networking*. 2018

Conference papers

— [99] Y. Desmouceaux, M. Enguehard, V. Nguyen, P. Pfister, W. Shao, E. Vyncke. "A Content-aware Data-plane for Efficient and Scalable Video Delivery". In *16th IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019

— [100] M. Enguehard, G. Carofiglio, D. Rossi. "A Popularity-Based Approach for Effective Cloud Offload in Fog Deployments". In *30th International Teletraffic Congress (ITC 30)*. 2018

— [101] M. Sardara, L. Muscariello, J. Augé, M. Enguehard, A. Compagno, G. Carofiglio. "Virtualized ICN (vICN): towards a unified network virtu-

alization framework for ICN experimentation". In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN)*. 2017

Workshop papers

— [102] M. Enguehard, D. Rossi, R. Droms. "SLICT: Secure Localized Information-Centric Things". In *Proceedings of the 3rd ACM Conference on Information-Centric Networking - Workshop on Information Centric Networking for 5G (IC5G)*. 2016

Posters and demonstrations

— [103] M. Enguehard, D. Rossi, R. Droms. "On the Cost of Secure Association of Information Centric Things". In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN)*. 2016

— [104] J. Augé, G. Carofiglio, M. Enguehard, L. Muscariello, M. Sardara, "Simple and efficient ICN network virtualization with vICN". In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN)*. 2017

— [105] J. Augé, M. Enguehard. "A network protocol for distributed orchestration using intent-based forwarding". In *16th IFIP/IEEE International Symposium on Integrated Network Management (IM) - Demonstration Session*

Talks and presentations

— *Information-Centric Things - Running ICN over RIOT*. RIOT Summit 2016. Berlin, Germany. 2016

— *ICN, IoT, and the FOG: enabling computation at the edge*. INRIA RIOT Seminar. Paris, France. 2017

— *Configuration, management and control of a CICN network demo*. ICNRG Interim Meeting. Prague, Czech Republic. 2017

— *Virtual ICN-based IoT networks with vICN and RIOT*. RIOT Summit 2017. Berlin, Germany. 2017

— *Tutorial: Community Information-Centric Networking (FD.io/cicn)*. 4th ACM Conference on Information-Centric Networking (ICN). Berlin, Germany. 2017

— *Objets centrés sur le contenu: application des réseaux centrés contenus à l'Internet des objets*. La Recherche montre en main (Weekly live science show on French public radio France Culture). 6 December 2017

— *Réseaux intelligents pour les villes intelligentes*. Journée TEDonnées. Besançon, France. 2018

Patent applications

— A. Compagno, L. Muscariello, G. Carofiglio, M. Enguehard. LIGHTWEIGHT NAMING SCHEME FOR AGGREGATING REQUESTS IN INFORMATION-CENTRIC NETWORKING. USPTO Application Number 15/943,775

— M. Enguehard, G. Carofiglio, D. Rossi. POPULARITY-BASED LOAD-BALANCING FOR FOG-CLOUD PLACEMENT. USPTO Application Number 16/043,550

— M. Hawari, Y. Desmouceaux, M. Enguehard, A. Augustin, A. Surcouf. RESILIENT TRANSMISSION OF RAW VIDEO STREAMS OVER AN IP COMMUNICATION NETWORK. USPTO Application Number 16/033,112

— M. Enguehard, Y. Desmouceaux, J. Augé. SYSTEM AND METHOD FOR MIGRATING A LIVE STATEFUL CONTAINER. USPTO Application Number 16/130,824

— M. Enguehard, J. Augé, G. Carofiglio, M. Papalini. DISTRIBUTION OF NETWORK-POLICY CONFIGURATION, MANAGEMENT, AND CONTROL USING MODEL-DRIVEN AND INFORMATION-CENTRIC NETWORKING. USPTO Application Number 16/178,967

— J. Augé, M. Enguehard, J. Samain, A. Compagno, M. Papalini. ANCHORLESS AND MULTI-RAT MOBILITY AND ROAMING MANAGEMENT. WTO Application Number PCT/US2018/049181

— M. Enguehard, Y. Desmouceaux, P. Pfister, M. Townsley, E. Vyncke. EFFICIENT AND FLEXIBLE LOAD-BALANCING FOR CLUSTERS OF CACHES UNDER LATENCY CONSTRAINT. USPTO Application Number 16/261,462

Chapter 2

Geographic routing for information-centric wireless sensor networks

In this first chapter, we focus on the retrieval of data from IoT networks. In particular, the problem of routing and forwarding in a multi-hop wireless sensor network (WSN) using ICN is addressed. Indeed, as noted in the introduction, recent research argues that ICN is a better fit for WSN than the standard IPv6-based network stack, For instance, seminal work [24] shows through experiments that a slightly modified ICN stack outperforms the standard IPv6-based stack (i.e., IEEE 802.15.4, 6LoWPAN and RPL) in terms of energy efficiency and memory requirements – which are primary concerns for WSNs.

However, the advantages resulting from the use of ICN architectures in terms of naming, mobility, and security (Section 1.2) do not come without challenges [36]. One such challenge is efficient packet delivery with minimal control traffic. Most of the recent pieces of work on ICN for the WSN rely on a flavour of flood-and-learn to discover network paths towards ICN names [24, 54, 55, 86, 91]. While simple from an implementation perspective, such approaches that use network broadcast can be costly for low-power devices, especially in mobile wireless networks where bandwidth is also a scarce resource [106].

2.1 Geographic routing

Geographic routing, where packets are routed towards a physical location instead of a host, has long been pushed as a potential solution for routing in WSNs [107]. Indeed, in typical geographic routing implementations, forwarding is performed by selecting the neighbour that is the closest to the packet's destination. Control traffic is thus kept local, as nodes only need to know their neighbours' position to take a forwarding decision. While geographic routing strategies have been thoroughly studied in the literature since the publication of seminal work such as [108], to date there are no implementations available in state of the art IoT stacks [109] and the same holds true for the ICN-WSN context.

In this chapter, we set ourselves to explore the feasibility and practicality of using geographic routing and forwarding in ICN-enabled WSNs, looking specifically at three aspects:

(i) how can geographic routing and forwarding be realized from a protocol point-of-view while respecting the ICN semantic?
(ii) how can the routing protocol be protected from rogue participants aiming at either diverting or denying requests?
(iii) how does geographic routing compare to the current flood-and-learn approach in terms of feasibility (memory and CPU footprint of the forwarding algorithm on the constrained devices) and efficiency (energy and network overhead of the routing scheme)? Rather than focusing on a specific WSN deployment, which would result in conclusions of limited scope, we intentionally study the broad issue of *geographic* as opposed to *name-based* routing and forwarding, where resource consumption grows respectively with the number of neighbours and entries in the ICN FIB.

Our contributions, matching the aforementioned challenges, can thus be summarized as such:

(i) we propose a protocol implementation of geographic routing in ICN that relies on a naming convention, a beaconing mechanism, and the ICN strategy layer to realize geographic forwarding. In particular, we implement a classic flavour of geographic forwarding, GPSR [108], in an ICN stack based on RIOT-OS [58].
(ii) security mechanisms for the protocol in (i) are introduced, in particular, to prevent rogue nodes to join the WSN and to emit beacons. A secure join protocol based on asymmetric cryptography is proposed and compared to a state-of-the-art protocol, OnboardICNg [68], itself based on symmetric cryptography. Trade-offs between the two protocols are presented, showing that OnboardICNg requires more network transmissions but less energy consumption.
(iii) we propose a simple analytical model to represent the energy consumption of WSN routing protocols based on either name-based or geographic forwarding; we use our protocol implementation to measure the protocol CPU footprint and gather accurate data from the literature on message encryption and transmission costs as one source of data for the model; we simulate message propagation dynamics over large topologies, from which we gather propagation patterns as the second source of data to the model; finally, given an energy budget, we use the model to derive the expected number of messages for different degrees of network dynamism under both schemes, giving useful guidelines for ICN-WSN deployments.

The content of this chapter has been the object of three publications. [102] covers the protocol implementation of geographic forwarding (i) and offers partial insights into its comparison to name-based approaches (iii), looking at the cost of forwarding a single packet. [103] contains the comparison between the two network join protocols (ii). [98] extends the work on comparing geographic and name-based forwarding, looking this time at the complete cost of the routing protocol (i.e., forwarding cost plus control plane cost).

The remainder of this chapter is organized as follows. We start by overviewing WSN deployments, with the purpose of selecting some relevant use cases

Table 2.1 – Reference WSN deployments

	Deployment name	Deployment class	No. of Nodes	Node degree	Ref.
A	Place de la Nation	Urban sensor network	97	3.8	[13]
B	Great Duck Island	Environmental sensor network	150	4.6	[110]
C	CASAS	Home automation	30	8	[4]
D	Sensor Andrew	Building automation	1000	15	[111]

that we use as reference points in our evaluation (Section 2.2). Next, we introduce the reference ICN architecture, discussing aspects related to naming, security, and forwarding (Section 2.3). We then formally state our problem and outline the methodology, introducing the energy model for ICN-WSN deployments at a high level (Section 2.4). We incrementally add details to the overall picture, refining each of the building blocks (Section 2.5–2.6). The full details of the model are then presented and used to quantitatively and qualitatively contrast geographic and named-based ICN forwarding for the identified use cases (Section 2.7). Finally, we discuss and summarize our main findings (Section 2.8).

2.2 Reference WSN deployments

Depending on the use cases and applications, WSN deployments cover a broad spectrum of characteristics in terms of node density, number of nodes, typical topology, traffic patterns, etc. Given that our main aim is to identify under which circumstances, if any, geographic forwarding is more advantageous than classical name-based forwarding, we need to select a number of *specific* use cases, representative of different application *classes*. For the purpose of *quantitative* assessment, we need each use case to precisely report characteristics that are specific to a single deployment. At the same time, provided that the selection is made among carefully defined application classes, we expect that the results for the selected example in any given application class *qualitatively* applies to other deployments in the same class.

As pointed out in Section 1.1, there are many classes of IoT applications. To better focus our investigation, we consider the IETF Routing Over Low power and Lossy networks (ROLL) working group [1], which identifies four main use cases: (i) urban sensing [12], (ii) industrial sensing [7], (iii) home automation [5], and (iv) building automation [2]. While not considered by the ROLL working group, environmental sensor networks and machine-to-machine deployments are another class of deployments largely covered in the literature [10,112]. Without loss of generality, and to give the reader several reference points, we consider four deployments, whose relevant characteristics we summarize in Table 2.1 with their classes with respect to the aforementioned ROLL categories. For each deployment, we review the reference documentation or available data to determine the number of neighbours for each node, which we use as input data

1. `https://datatracker.ietf.org/wg/roll/`

to our model. The deployments, ordered by increasing node degree in Table 2.1, are:

(A) **Place de la Nation**: as an example of the urban sensing class, we take the Cisco-Paris deployment, which is a joint venture between Cisco, the City of Paris and several start-up companies [13]. The deployment is used to measure and track car and pedestrian traffic and pollution patterns on a highly frequented square in Paris. It consists of 19 cameras, 14 noise sensors, 5 pollution-reading sensors, and 12 wireless access points that report information about user connections. The measured data is open-source and available online [113].

(B) **Great Duck Island**: the Great Duck Island deployment [110] is an environmental sensor network consisting of 150 devices. The sensors were used to observe the habitat of seabirds on an island off the coast of Maine and its evolution depending on weather conditions.

(C) **CASAS**: as an example of the home automation class, we select CASAS [4], which is a so-called "smart-home in a box": a ready-to-deploy sensor network that allows any consumer to transform their home in a connected (or "smart") home. It consists of 30 nodes communicating over the IEEE 802.15.4 radio channel, including temperature sensors and infrared motion/light sensors.

(D) **Sensor Andrews**: as an example of the building automation class, we select Sensor Andrew [111], a sensor network deployment at Carnegie Mellon University (CMU). More than 1000 devices spread all over the CMU campus report numerous measurements such as electricity consumption or temperature.

2.3 Reference Information-Centric Things (ICN-WSN) Architecture

To realize secure geographic routing, several building blocks must be added to tradition ICN-WSN architectures: (i) a *neighbour discovery* and association protocol (Section 2.3.1), which ensures that only trusted nodes are authorized to send packets on the network; (ii) a *secure beaconing* (Section 2.3.2) protocol to handle topology and location changes; (iii) a *forwarding scheme* (Section 2.3.3), to ensure correct forwarding of Interest packets over the network independently of the forwarding algorithm class (i.e., geographic or name-based). In this section, sample implementations for these are detailed and discussed, highlighting the various trade-offs. Let us note that our study leaves for future work one important feature of ICN: in-network caching. As memory is a scarce resource on IoT platforms, a better understanding of the opportunities for in-network caching in the ICN-WSN would require a thorough study of current ICN-WSN stacks and their memory usage. We provide a first step in that direction with the results of Section 2.7.2.

2.3.1 Secure neighbour discovery

Whereas in the context of fixed ICN networks, security is attached to self-verifiable data objects, the world of ICN-WSN requires additional features. To

begin with, given the broadcast nature of the wireless medium, in a hostile environment silent attackers could eavesdrop on sensitive sensor data. Additionally, given the multi-hop nature of WSN communications, talkative attackers could instead swamp network resources, such as battery and wireless medium, by issuing bogus Interest messages. Additional security mechanisms are thus required, such as naming and communication patterns to enforce access control on ICN-based WSNs [67, 68].

This section thus addresses the problem of neighbour discovery and association for ICN-WSN, to ensure that only trusted nodes are authorized to send packets on the wireless network. A novel association protocol based on asymmetric keys is presented and compared to a recently proposed one based on symmetric cryptographic keys [68]. The evaluation considers both security and network properties of these protocols, as well as important practical aspects such as the forecasted power consumption of the protocol implementation on different WSN technologies.

Association protocols using (a)symmetric cryptography

Symmetric Cryptography. Let us first note that, while symmetric cryptography is not natively suited to authentication, it is several orders of magnitude less expensive in terms of CPU cycles than standard asymmetric cryptography [114], which makes it attractive for low-power environments. Additionally, there is a growing hardware support for symmetric cryptography on recent sensor boards, which implies a shrinking energy footprint of cryptographic operations. This explains why the author of OnboardICNg [68], an ICN-based neighbour authentication protocol, selected to rely only on symmetric cryptography. To overcome the limitations of symmetric cryptography, OnboardICNg requires nodes to be pre-configured with a secret symmetric key shared with a central authority, which must then be contacted at each authentication exchange and acts as a trusted third party between the nodes in the ICN-WSN. In particular, if d_j is a node trying to join the network and d_{nbr} is a neighbour in charge of the authentication, an OnboardICNg exchange yields the following results:

1. d_j proves to d_{nbr} that it shares a symmetric key with the central authority;
2. symmetrically, d_{nbr} proves to d_j that it shares a symmetric key with the central authority;
3. d_{nbr} generates a secret symmetric key for d_j and d_{nbr} to encrypt their unicast layer-2 traffic;
4. d_{nbr} and d_j use the aforementioned key to exchange broadcast keys (the broadcast key is a symmetric key propagated by one node to its direct physical neighbour to enable encrypted L2 broadcasts).

Asymmetric Cryptography. We further design an ICN-based protocol that uses asymmetric cryptography, such as Elliptic Curve Cryptography (ECC). Compared to symmetric cryptography-based OnboardICNg, where every authentication session requires contact with an authentication server, asymmetric cryptography allows nodes to authenticate each other without any third party. Local exchanges imply spatial reuse of the wireless medium and reduce the energy footprint due to relaying traffic towardsthe authentication server. This

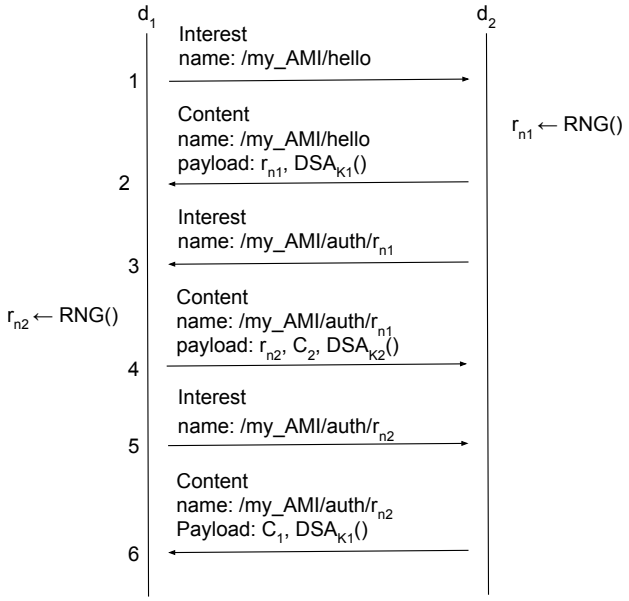


Figure 2.1 – ICN-based protocol for authentication using asymmetric cryptography

is especially critical for nodes close to the authentication server that are more solicited and whose battery would be quickly depleted. At the same time, while asymmetric cryptography is rather commonly used in association with a public key infrastructure to perform authentication (e.g., in TLS [115]), it requires computationally expensive operations, which may not be a good fit for energy-constrained nodes.

Given that the design of an asymmetric cryptography protocol with the same security properties as the symmetric cryptography-based OnboardICNg [68] is an original contribution of this work, we briefly sketch its inner working in fig. 2.1. In this scheme, each node d_i has a pair of asymmetric keys K_i, with its corresponding certificate C_i signed by a trusted third party (e.g., the authentication server). Signing a message with a key K_i is noted as $DSA_{K_i}()$ and $RNG()$ is a random number generation function. To authenticate itself, a node must prove that it owns a key that has been certified by the trusted third party (messages 4 and 6). The nonces r_{n1} and r_{n2} protect the protocol against replay attacks by providing a challenge-response authentication. They can also be used to derive a symmetric session key, for instance with the Diffie-Hellman algorithm.

Network vs Energy Footprints

We estimate the footprint using two sensors, the older TelosB (with 16-bit MSP430 CPU) and a new-generation OpenMote (with 32MHz ARM Cortex-M3 CPU). We consider respectively ECC160 for asymmetric cryptography and AES-CCM-128 for symmetric cryptography. Interestingly enough, the Open-

Table 2.2 – Cost of encrypting (AES-128) or signing (ECC) 128b on the TelosB and OpenMote

TelosB		**OpenMote**	
ECC160 sw	AES128 hw	ECC192 sw	AES128 hw
15 mJ [116]	14.3 µJ [117]	11.4 mJ [118]	0.9 µJ [118]

Table 2.3 – asymmetric cryptography vs symmetric cryptography-based authentication protocols

Board	**Crypto**	**Messages (#)**	**Energy (mJ)**	**Latency (s)**
TelosB	AES hw	9	4.3 – 6.4	1.4
	ECC sw	6	53.3 – 57.3	10.9
Open	AES hw	9	0.54 – 0.89	0.13
Mote	ECC hw	6	22.5 – 28.7	0.95

Mote supports both AES and ECC in hardware. We collect energy costs of cryptographic operations in table 2.2, which we use for the performance evaluation. We then contrast (i) number of messages, (ii) energy cost and (iii) latency for both schemes in table 2.3.

On the one hand, we observe that only 6 messages are required in asymmetric cryptography compared to 9 in OnboardICNg – a 30% reduction. Additionally, exchanges in the asymmetric cryptography case are confined to neighbouring devices, whereas in the symmetric cryptography case messages need to reach a sink point (the authorization entity). Hence, not only does asymmetric cryptography requires fewer messages, but these messages have a shorter delay and involve fewer hops in the network. These are all desirable properties that make asymmetric cryptography an interesting alternative to symmetric cryptography-based protocols such as OnboardICNg [68]. On the other hand, we also gather that cryptographic functions dominate latency overhead for asymmetric cryptography — by about 8x. Similarly, energy-wise the performances are largely favourable to OnboardICNg — up to 41x. It must be noted however that the cost of transmitting messages reduces the performance gap between asymmetric cryptography and symmetric cryptography shown in table 2.2 where ECC-160 requires up to 10^4 more energy.

Given this very large performance gap, it follows that the advantages in terms of the network communication cost are completely offset by the large penalties in terms of latency and energy. This finding leads to an interesting trade-off, depending on the characteristics of the WSN: on the one-end, symmetric cryptography is less energy- and latency-consuming; on the other hand, it requires connectivity to a trusted third party even though the low-power wireless technologies used in WSN are subject to temporary network partitions due to, e.g., interference or duty-cycling. Thus, while symmetric cryptography is better suited for authentication in dynamic WSNs with reliable connectivity (e.g., dense and mobile networks such as smart cities), asymmetric cryptography is better suited to static networks with intermittent disconnections (e.g., environmental sensor networks). As mobility is one of the challenges we set ourselves to solve in this chapter, OnboardICNg [68] is selected as the reference authentication protocol.

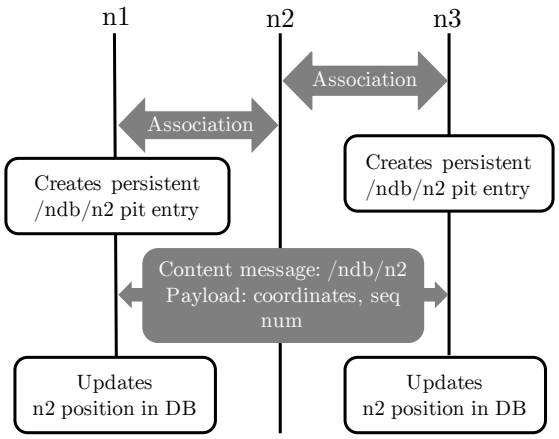


Figure 2.2 – Synoptic of the ICN-WSN beaconing protocol

2.3.2 Secure beaconing

Beaconing presents two new challenges. First, *unsecure beaconing* opens the possibility of wormhole or DoS attacks through exhausting the neighbour database or overloading the central processing unit (CPU). Second, beaconing is essentially a *push* operation, which contrasts with the ICN *pull* model.

Security. In order to prevent these threats, sensors must be able to distinguish between beacons originating from trusted and malicious entities. We thus use the broadcast keys provided by OnboardICNg [68] to encrypt our beacons and authenticate their origin. All the subsequent messages are encrypted with the node broadcast key and contain a message authentication code (MAC). Using this encryption, ICN-WSN devices are resistant to flooding attacks from non-authorized nodes. Indeed, only beacons encrypted with the broadcast key of authenticated neighbours are considered, and the corresponding key can only be accessed by trusted nodes. However, the scheme is not resistant to trusted nodes that have been physically tampered with.

Note that if the AES operations have to be performed in software, attackers can send packets with bogus encryption to perform a simple DoS attack against a node's CPU. However, recent platforms such as the OpenMote Section 2.4.1) are equipped with hardware modules that can perform AES computation. Therefore, these systems compute and check MAC at low CPU and energy cost.

Push. To accommodate the push nature of beacons with ICN, we must slightly modify the specification of ICN exchanges, similar to the work presented in [119]. Specifically, we use *persistent Pending Interest Table (PIT) entries* (i.e., entries that are not purged after being satisfied once) and *unsolicited Data messages* (i.e., Data messages that are emitted without a corresponding Interest message). We describe the beaconing protocol with the help of Figure 2.2:

(i) After an OnboardICNg association, each node creates a persistent PIT entry (e.g., with a soft timeout) for `/ndb/neigh_id`, where `neigh_id` is the id of the neighbour with whom the exchange was performed.

(ii) Regularly, each node sends a broadcast unsolicited Data packet (encrypted

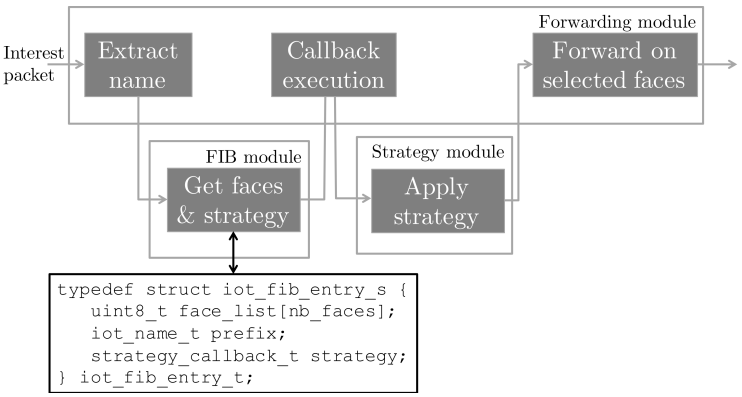


Figure 2.3 – Coexistence of forwarding strategies in the reference ICN-WSN stack

with the node broadcast key) for `/ndb/node_id` containing the beacon information (e.g., the node's coordinates) and a sequence number (to avoid replay attacks).

(iii) Unsolicited Data packets are forwarded to the beacon processing application, thanks to the persistent PIT entry.

Persistent PIT entries and unsolicited messages have a network utilisation advantage over the traditional ICN Interest/Data exchange. Indeed, the traditional scheme requires four packets per pair of neighbour nodes (two exchanges, one per node), so a total of $4Nd$ where N is the total number of nodes and d the average number of neighbours per node. Instead, with our scheme, each beacon is broadcast to all of the nodes in the neighbourhood, so that only N packets are required.

Deriving from the standards ICN specifications comes at a cost as the pull model is central to many aspects of the ICN architecture. One such example is congestion control [120] where the Interest-to-Data balance is used to perform both receiver-based interest pacing and packet scheduling at intermediate nodes [121]. Another is protecting the network against DoS attacks (specifically reflection and bandwidth depletion attacks as defined in [122]) by ensuring that nodes do not receive Data packets that they have not requested with a corresponding Interest. Both of these issues can, however, be mitigated for the geographic beacons presented in this section:

— beacons are purely link-local and never routed and are thus not subjected to congestion control or packet scheduling (similarly to, e.g., IPv6 Neighbour Discovery [123]); to enforce this property, the implementation of the ICN stack should prevent creating persistent PIT entries that are not associated with an application face;
— beacons are also encrypted using the broadcast key distributed during the association protocol presented in Section 2.3.1; thus, only authenticated and trusted nodes are allowed to issue unsolicited data packets.

2.3.3 Forwarding

Our reference ICN-WSN architecture is conceived as a framework to perform name-based and geographic forwarding in the ICN-based WSN and is thus independent of the actual variation of geographic forwarding chosen. We achieve this using the strategy layer introduced in Section 1.4.1 with the workflow summarized in Figure 2.3. In our ICN stack implementation, FIB entries match with faces and strategies. Faces can be either physical neighbours, application or virtual faces (such as the broadcast face). A strategy is a *callback* on the faces in the FIB, that can, for instance, select a face among the available ones with a specific metric. For instance, one could use a specific prefix (such as `/g/`) to forward packets through geographic forwarding by linking it to the corresponding strategy in the FIB. Interest packets destined to any other prefix would still be forwarded by name.

While our architecture allows for a variety of forwarding strategies, for quantitative performance evaluation we need to select specific geographic and name-based forwarding strategies that are implemented and executed on real IoT hardware. Routing and forwarding in the WSN world have been the subject of extensive research; we refer the reader to [124] for a taxonomy and survey of algorithms. To guide our selection, we remark that this taxonomy, which categorizes forwarding into flat/hierarchical or location-based strategies, also applies in an ICN-flavoured WSN, where the choice of using flat/hierarchical or location-based *naming schemes* directly maps the choice of a forwarding strategy as well [125].

Location-based. To select our candidate location-based strategy, we remark that most geographic forwarding techniques are based on greedy forwarding (i.e., select the neighbour closest to the destination as a next hop) with either a beacon-based [108, 126] or beacon-less [127, 128] approach. Greedy choices are complemented by recovery techniques to route around sinkholes, as in GPSR [108] or GOAFR+ [126].

The applicability of geographic forwarding to ICN has explored in a limited way, primarily as applied to Vehicle-to-Vehicle (V2V) networks [91,129,130] and are designed to exploit V2V characteristics: highly dynamic, fast moving nodes with no battery/CPU constraints that receive long streams of video/audio data. In [131], the authors propose an ICN-WSN routing scheme based on geographic coordinates. However, their proposal assumes a tree-like topology and does not account for potential sinkholes. We additionally note that, despite the availability of many geographic forwarding algorithms in the literature, there are few implementations; e.g., even the most basic and best-known approaches, such as GPSR [108], are not available in modern IoT toolboxes [109] such as Contiki [132] or RIOT [58]. As a representative of location-based strategies, we implemented GPSR [108], a classic and well-understood strategy based on a geographic greedy forwarding algorithm. The implementation of GPSR is made available as open-source [133].

In addition to destination coordinates (that are part of a name under the `/g/` prefix), GPSR also requires additional information for the forwarding. Indeed, to avoid local maxima (cases where the current node is closer to the destination than any of its neighbours), GPSR uses a technique called "perimeter routing", which requires the packet to carry the coordinates of the node where it entered

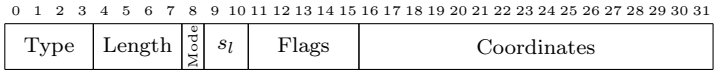


Figure 2.4 – The TLV field used by GPSR for keeping in-flight state (with s_l=16).

the perimeter mode. The ICN-WSN architecture stores this information in a Type Length Value (TLV) field as described in Figure 2.4, where a flag determines whether the GPSR is in greedy or perimeter mode. Given that we expect the reference ICN-WSN architecture to be used in different scenarios (dense deployment in urban buildings as opposed to sparse deployments in large rural areas), it may be desirable to provide the capability to adjust the coordinate resolution to a specific application scenario to avoid overhead. As a result, the architecture also supports the use of different resolutions for geographic coordinates: as Figure 2.4 outlines, 2 bits in the flags, noted as s_l, allow to specify from 8 to 64 bits coordinates, in step of 8 bits.

Name-based. The matching between Interest names and output faces in the ICN-WSN FIB is usually done using longest-prefix match. Simple flood-and-learn (F&L) forwarding strategies [24, 134] (and variants [135, 136]) are typically used in ICN-WSN to construct FIBs on demand. These are inherently non-scalable and possibly a bad fit for IoT deployments. As such, while we do implement F&L for reference purposes, we do also argue for the need to complementing it with additional techniques to reduce broadcast storms. Particularly, we opt for a Multi-Point Relay (MPR) [137] variant inspired by the Optimized Link State Routing Protocol (OLSR), where at each hop during the message propagation, only a few relays are selected out of those having received the message.

However, while implementing naive F&L is simpler than implementing GPSR, implementing a full-blown MPR distributed OSLR-like protocol is not. In particular, F&L and MPR performance will not only differ in the number of messages sent over the network (which we can simulate) but will also differ in the computational complexity (which we should measure from an actual implementation). We can approximate the computational cost of MPR with the computational cost of the simpler F&L strategy to provide a conservative evaluation of the energy efficiency of MPR for comparison with GPSR. Therefore, we prototype only the simpler F&L for the purpose of measuring the computational cost in Section 2.4.3, and simulate an ideal MPR for the purpose of gathering a lower bound of the number of messages transmitted by MPR over a network in Section 2.6.2. In particular, our ideal MPR implementation exploits global knowledge available in simulation to find a minimal set of MPR relays, providing a lower bound on the message complexity. These choices guarantee a conservative evaluation of the potential benefits of geographic GPSR over name-based MPR.

2.4 Methodology overview

In this section, we define the lines along which we evaluate the costs of using name-based and geographic forwarding strategies in ICN-WSN deployments. In

Table 2.4 – Characteristics of the OpenMote

Architecture	ARM Cortex-M3 (32 bits)
MCU	Texas Instrument CC2538 (32 MHz)
RAM (ROM)	32 kB (512 kB)
Encryption HW	AES & ECC
Encryption cost [118]	19.7 µJ (SW, AES-CCM, 128bits) 8.7 µJ (HW, AES-CCM, 128bits)
Consumption [138]	39 mW (CPU at 32 MHz, no RX/TX) 60 mW (CPU idle, RX at -50 dBm) 72 mW (CPU idle, TX at 0 dBm)

particular, we detail the experimental setup (Section 2.4.1) and focus our attention on the most relevant implementation criteria: namely, (i) *memory footprint* (Section 2.4.2), (ii) *CPU overhead* (Section 2.4.3), and (iii) *energy consumption* (Section 2.4.4). We consider memory and CPU metrics as tied to the *feasibility* of an ICN-WSN deployment, whereas we use the energy consumption to quantify the *cost* of ICN-WSN operation. We found that assessing memory and CPU costs is significantly simpler than assessing the overall energy budget, which, therefore, is the main contribution of our investigation. We thus set to solve the following problem: *under which conditions (if any) is geographic forwarding more performant in terms of energy, memory, and CPU consumption than flood-based strategies for the ICN-WSN?*

A complex system such as a WSN deployment is influenced by numerous factors (that are recapped and summarized further on in Table 2.6). Therefore, we employ a range of methodologies. At each step of our evaluation, we select the most appropriate one to estimate values that have practical relevance for the different variables. This section defines our ICN-WSN experimental model and enumerates the various sources of energy consumption. Sections 2.5 and 2.6 combine measurement from an actual ICN-WSN implementation with stochastic modelling of L2 transmission and simulation of network-wide scenarios to populate the different components of the model, including security, forwarding, data plane traffic, and control plane overhead. The refined model is then quantitatively analysed to provide guidelines on the most favourable ICN-WSN settings for the different WSN deployment classes and scenarios (Section 2.7).

2.4.1 Experimental setup

While our methodology to evaluate the cost of secure geographic forwarding in the reference ICN-WSN architecture is general, the quantitative aspects reported in this chapter are relevant for the hardware and software setup with which we conducted our evaluation. To make them of interest to the largest possible audience, we selected widely used open-source hardware (OpenMote [139]) and software (RIOT OS [58]) stacks.

Hardware setup. The OpenMote platform has a 32 MHz ARM Cortex-M3 CPU and is equipped with an IEEE 802.15.4 chipset as well as hardware modules for symmetric and asymmetric cryptography. To evaluate the cost of hardware cryptography and of receiving or transmitting packets through the IEEE

Table 2.5 – Size of Interest (I), Geo-interest (GI), and Data (D) ICN frames

			Packet Type		
	Field	**Field size**	**I**	**GI**	**D**
L2 header	802.15.4 PHY header	6B	✓	✓	✓
	802.15.4 MAC header	23B	✓	✓	✓
	802.15.4 SEC header	5B	✓	✓	✓
L3 header	Packet Type TL	1B	✓	✓	✓
	Nonce TLV	1B (TL) + 1B (V)	✓	✓	
	Name TL	1B	✓	✓	✓
	Name component TLVs	s_n	✓	✓	✓
	GPSR TLV	1B (TL) + $(1 + s_l)$ (V)		✓	
Payload	Content TLV	1B (TL) + s_c (V)			✓
	Signature Info TL	1B			✓
	Signature Type TLV	1B (TL) + 1B (V)			✓
	KeyLocator TLV	1B (TL) + 1B (V)			✓
	KeyId TLV	1B (TL) + 1B (V)			✓
	Signature TLV	1B (TL) + 16B (V)			✓
Footer	802.15.4 Signature	16B	✓	✓	✓
	802.15.4 CRC	2B	✓	✓	✓

Total size	**Packet Type**
56B + s_n	Interest
58B + s_n + s_l	Geo-Interest
79B + s_n + s_c	Data

802.15.4 interface, we rely on measurements performed by Shafagh et al. [118]. The energy consumption figures for this platform are provided in the corresponding datasheet [138], which we summarize along with other characteristics in Table 2.4.

Software setup. Our code runs on top of the RIOT operating system [58]. We implement a custom ICN stack on top of RIOT that uses standard ICN forwarding (i.e., longest-prefix match in the FIB) as well as GPSR (with perimeter routing as introduced earlier). To accommodate the typically low frame sizes of WSN (e.g., 127 bytes for IEEE 802.15.4 networks), adaptations to the TLV-based format of ICN packets have been proposed. Following the recommendations in [140], we implement 1+0 TLVs (i.e., where the Type and Length field are encoded in one single byte), instead of the 1+1 or 2+2 format described in the CCNx specifications [141], with which our implementation is otherwise fully compliant. Table 2.5 details the different fields of IEEE 802.15.4 ICN Interest, geographic-Interest and Data frames and reports the total frame size (as a function of the name or location size). As shown by this table, the geographic-Interest packet format differs from the ICN Interest only by the presence of the GPSR TLV that is used on top of the name and name-components TLVs to perform the routing.

2.4.2 Memory

Memory is a primary constraint in WSN. For example, an old platform such as the MSP430-based TelosB only offers 10KB of random-access memory (RAM) and 48KB of flash memory. Even recent hardware like the OpenMote includes only 32KB of RAM and 512KB of flash memory. This amount is still tiny considering that recent implementations of an ICN-WSN stack require already between 5KB [24] and 11KB [62] of RAM. Additionally, optimizing memory consumption is especially interesting in the context of ICN, where caching can be used to accommodate nodes with low duty-cycles [142].

When considering memory requirements, geographic forwarding has advantages over name-based forwarding. Indeed, under geographic forwarding the size of the state retained by a node to be able to forward any packet is bounded by the node degree, whereas under F&L (and variants) each node needs to retain some state for the name of every other node. In fixed ICN networks, state explosion is alleviated by using prefix aggregation in the FIB. However, this is hardly possible in highly-dynamic and mobile WSNs, and to the best of our knowledge no aggregation scheme has been proposed for ICN applied to WSNs.

Memory requirements can be computed considering that under geographic forwarding, the FIB contains the coordinates (having size s_l) of all its d neighbours, and that the beaconing protocol described in Section 2.3.2 additionally requires nodes to store a persistent PIT entry (having size s_{pit}) for each of their neighbours. Under classic name-based forwarding, ICN requires having one FIB entry (having size s_{fib}) for each of the n_s reachable names. As no aggregation scheme is currently available for dynamic WSN topologies, n_s is equal to the number of nodes in the WSN. This means that each node has a FIB entry for every other node in the network. To compute the required memory, it must be noted that both FIB and PIT entries also contain a 1-byte pointer to an ICN face and that a FIB entry also contains a 1-byte pointer to a strategy. We can thus express the respective memory requirements as:

$$\begin{aligned} M_{geo} &= d(s_l + s_{pit}) = d(s_l + s_n + 1) \\ M_{fib} &= n_s \times s_{fib} = n_s(s_n + 2) \end{aligned} \tag{2.1}$$

2.4.3 Computation

CPU power is another strong constraint on IoT platforms: this holds for both old platforms such as TelosB (16-bit CPU clocked at 8 Mhz), as well as for newer platforms such as the OpenMote (32-bit CPU clocked at 32MHz). An inefficient forwarding algorithm on a slow processor can delay message forwarding, causing congestion in the network.

We remark that the CPU complexity of a forwarding algorithm is also inherently dependent on the underlying hardware: for instance, multiplication on 32-bits integers is much faster on newer 32-bit CPUs than older 16-bit ones. It is thus only possible to evaluate the strength of a forwarding implementation with respect to a specific platform. More specifically, we can evaluate the number of CPU cycles $n_c(algo)$ required in a given assembly language for a specific implementation of any given strategy and then compute the corresponding energy E_{CPU} consumed by the CPU given its frequency f_{CPU} and its power drain

Listing 2.1 – Benchmarking code

```
uint32_t do_iteration () {
        //Initializes structures and counter
        do_initialize ();
        DWT->CYCCNT = 0;

        //Performs the micro-benchmark
        perform_bench ();

        //returns the number of used CPU cycles
        return DWT->CYCNT;
}
```

P_{CPU} from data-sheets:

$$\begin{aligned} E_{CPU}(algo) =& P_{CPU}\, t_{CPU}(algo) \\ =& P_{CPU} \frac{n_c(algo)}{f_{CPU}} \end{aligned} \tag{2.2}$$

To devise an accurate model, we need a method to reliably measure the number of cycles $n_c(algo)$. Given that CPU emulators or static code analysis are subject to low accuracy [143], we opt for *micro-benchmarking* the different pieces of the reference ICN-WSN architecture code with cycle-level accuracy, using a simple yet powerful technique. To accomplish micro-benchmarking, we use a special register of the Cortex-M3 CPU dedicated to counting CPU cycles [2]. This register is directly mapped in memory and can be accessed on RIOT through the `DWT->CYCCNT` variable, without performance penalty. An example of the micro-benchmark code is presented in listing 2.1.

2.4.4 Energy

The energy cost of a specific ICN-WSN implementation comes from three main sources: a *computational cost* related to the forwarding algorithms, a *security cost* related to cryptographic operations and message exchanges due to the security protocol, and a *network cost* related to point-to-point communication, end-to-end transmission, and network maintenance.

During the overall lifetime of an ICN-WSN deployment, network cost can be split into bootstrap, forwarding of Interest/Data packets, and handling of route failures, all of which are clearly dependent on the forwarding strategy employed. Network bootstrap is the cost of setting up the forwarding for the full network to be able to forward packets from any node to any other. Once routes are set up, communication under different forwarding strategies incurs different costs. Indeed, the amount of energy spent for forwarding depends on the computational cost of the forwarding algorithm, on communication costs because of additional state embedded in the Interest packets, and on the different numbers of relays under each algorithm. Finally, handling route failure is an operation common to volatile environments such as WSN deployments, where routes to content can become unavailable due to mobility or poor channel conditions. Reacting to this failure triggers the (re)discovery of a path, a costly operation that must be considered and whose cost depends on the forwarding strategy.

2. The CYCCNT register, see `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0337e/ch11s05s01.html`

In our evaluation, we assume that network load is uniformly distributed over the ICN-WSN network, i.e., that each node forwards probabilistically the same amount of traffic. This simplifying assumption, necessary for the tractability of the model, is justified by the fact that we consider networks with either *user/sensor mobility* or *machine-to-machine communications*, which tend to make the existence of a single hotspot sink less prevalent. Furthermore, we do not aim at precise absolute evaluation of the network energy spending in a specific scenario but at a relative comparison of name-based and geographic forwarding for the ICN-WSN. We can then compute the total network cost by summing the $E_{bootstrap}$, $E_{forwarding}$ and E_{change} network-related components as follows:

$$\begin{aligned} E_{total} &= E_{bootstrap} + N_m E_{forwarding} + \frac{N_m}{f_c} E_{change} \\ &= N_m E_{forwarding} + (1 + \frac{N_m}{f_c}) E_{change} \end{aligned} \tag{2.3}$$

where N_m is the number of useful ICN queries (i.e., expression of Interest satisfied by a Data packet in a multi-hop fashion) and f_c is the number of queries performed between two route changes. Intuitively, f_c represents the level of dynamism in the network, which lumps altogether phenomena such as (i) the addition/removal of new names or sensors, (ii) mobility of physical devices, (iii) route failure due to the wireless medium. Arguably, $E_{bootstrap}$ is a one-time cost with vanishing impact over time, so we approximate it with $E_{booststrap} \approx E_{change}$, i.e., the control-plane cost of repopulating a FIB. $E_{forwarding}$ instead represents the energy cost of a single query in the WSN once the FIB is already populated, and as such accumulates the data-plane costs to transmit an Interest over several hops in the network as well as the cost of receiving the corresponding Data packet travelling in the opposite direction. $E_{forwarding}$ and E_{change} are addressed in Section 2.5 and Section 2.6 respectively.

Finally, it must be noted that the number of messages N_m is not a design parameter. Rather, we can re-express Equation (2.3) to infer the total number of exchanges N_m that are possible, as a function of the network dynamism f_c, under different forwarding strategies:

$$N_m = \frac{E_{total} - E_{change}}{E_{forwarding} + \frac{1}{f_c} E_{change}} \tag{2.4}$$

Without loss of generality, in Section 2.7 we exploit Section 2.4.4 where we equate the total energy budget to the amount of energy available in standard AA batteries, i.e., $E_{total} = E_{AA}$.

2.5 Cost of forwarding a single ICN packet

In this section, we set out to evaluate the energy spent by a node to forward a single ICN packet as the first refinement of our energy model. We then evaluate this model for the OpenMote, using our own experiments and data gathered from the literature. From the point of view of a relay, the packet forwarding process can be divided into 5 steps, namely: (i) frame reception, (ii) frame decryption, (iii) forwarding face selection through the forwarding strategy,

Table 2.6 – Summary of variables used in the evaluation

Parameter	Symbol	Default value
No. of neighbours	d	15
No. of ICN names	n_s	2000
No. of FIB entries	n_f	15
Size of a location info	s_l	8 B
Size of a name	s_n	$\lceil \log_2(n_s) \rceil$
Size of the content	s_c	32 B
Energy cost of AES encryption	E_{AES}	10 µJ
No. of tries / transmission	$n_{tr,s}$	eq. (2.6)
L2 drop probability	p_c	[144]
No. of times a packet is forwarded during flood (including L2 retries)	$N_{tr}(T, D)$	eq. (2.19)
Energy cost of transmission/bit	E^b_{tx}	1.163 µJ
Energy cost of reception/bit	E^b_{rx}	0.96 µJ
Max number of hops of a packet on the WSN	T	8
Size of an ICN Interest packet	s_i	56B + s_n
Size of a geographic ICN packet	$s_{i,g}$	58B+s_n+s_l
No. of Interest/Content exchanges before a route change	f_c	free parameter
Energy content of an AA battery	E_{AA}	15 390 J
Budget of Interest/Data exchanges during the lifetime	N_m	eq. (2.21) and eq. (2.20)

(iv) frame encryption, and (v) frame transmission. We summarize the various variables used in the mode in Table 2.6.

2.5.1 Frame transmission and reception

The transmission and reception cost of an ICN packet is given by the amount of time that the node's antenna has to be powered in transmission (TX) and reception (RX) mode. Since the power consumption P_{tx} (resp. P_{rx}) of the platform is dependent on the hardware and available from the data sheets provided by the manufacturer, only the transmission time needs to be computed.

At any hop, the transmission time is driven by two factors: the number of retransmissions that are necessary for a successful reception on the wireless medium and the size of the message (which impacts the time taken by each retransmission). Let $n_{tr,s}(p_c)$ be the average number of tries necessary for successful transmission on a channel with collision probability p_c and capacity C_{phy}. The transmission cost of a frame of size s_f is then given by:

$$E_{tx}(s_f) = P_{tx} \frac{s_f}{C_{phy}} n_{tr,s}(p_c) = E^b_{tx} s_f n_{tr,s}(p_c) \tag{2.5}$$

The reception cost can be similarly derived.

Let M_{tr} be the maximum number of L2 retransmissions of a given frame (after which the frame is dropped). For a given p_c, we have that $n_{tr,s}(p_c)$ is, in

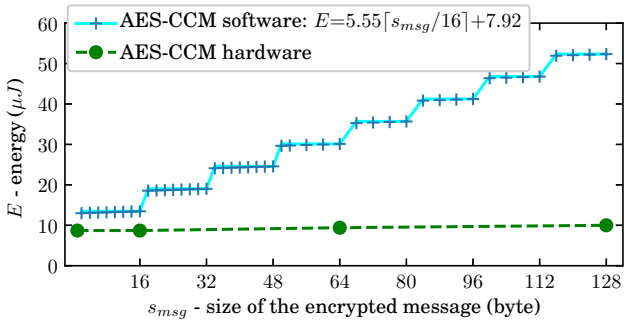


Figure 2.5 – Energy cost of AES-CCM based on size of message. Software encryption measurement are gathered as described in Section 2.4.3. Measurement for hardware encryption are provided in Table 6 of [118]

expectation:

$$\begin{aligned}\mathbb{E}[n_{tr,s}(p_c)] &= \sum_{k=1}^{M_{tr}} kP(k \text{ L2 transmissions needed}) \\ &= \sum_{k=1}^{M_{tr}} kp_c^{k-1}(1-p_c) \\ &= \frac{1-p_c^{M_{tr}}(M_{tr}+1)+p_c^{M_{tr}+1}M_{tr}}{1-p_c}\end{aligned} \tag{2.6}$$

We stress that p_c is not a system parameter but depends in turn on other properties of the ICN-WSN deployment, such as node density and radio range. We come back to p_c in Section 2.6.

2.5.2 Data Encryption and Decryption

The cost of cryptography depends on the device's capabilities, such as CPU characteristics, and more importantly on the availability of hardware cryptography components. We present the energy consumption of AES-CCM encryption in the OpenMote platform, considering both software- and hardware-assisted cryptography as a function of the message's size in Figure 2.5.

For the software implementation, we microbenchmark the AES implementation of the `crypto` module of RIOT with the previously outlined technique. Figure 2.5 provides the measurements as well as an equation derived from the measurements. For the hardware-assisted encryption, we point out that cryptography hardware modules are not yet supported on RIOT: we thus use as a reference the AES-CCM measurement reported in Table 6 of [118]. The picture clearly shows the importance of hardware modules for cryptographic operations: the AES-CCM software implementation consumes up to 5 times more energy than its hardware-assisted counterpart. In hardware, AES-CCM has a maximum cost of 10 µJ per packet (since the IEEE 802.15.4 maximum transmission unit (MTU) is 127B).

In the rest of the chapter, we assume that the platform is equipped with a hardware module that is accessible through an API of the software stack (as

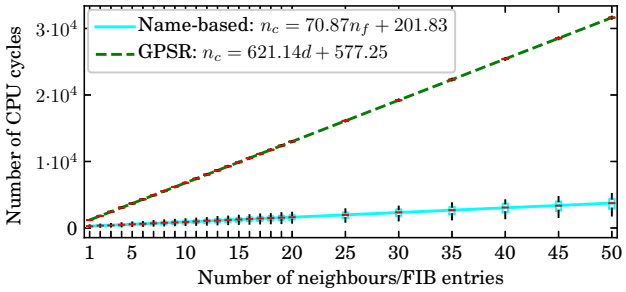


Figure 2.6 – Cycles per forwarding decision on the OpenMote based on number of neighbours/FIB entries. Boxplots report cycle-level accurate measurements with the method in Section 2.4.3, lines reports linear fitting of the data.

planned in RIOT). With hardware-assisted encryption, as Figure 2.5 shows, we can assume that costs for encryption/decryption are constant with respect to the frame size. Finally, given that encryption/decryption costs are not the main components of packet transmission, for the sake of simplicity, we assume that encryption and decryption have an equal cost, which we indicate with E_{AES}.

2.5.3 Forwarding algorithm

Deducing the cost of the forwarding algorithm is fairly simple: Equation (2.2) requires a microbenchmark of the forwarding code to accurately measure the number of CPU cycles required to perform either name-based ICN forwarding (i.e., longest-prefix match in the ICN FIB) or GPSR forwarding operations. Note that in this section, we do not yet consider the control traffic, but the name-based FIB would typically be populated using F&L or MPR. These measurements are reported in Figure 2.6, which shows boxplots of the number of CPU cycles as a function of the number of entries in the FIB (for name-based ICN) or the number of neighbours of the node (for GPSR) in the x-axis. Incidentally, the figure shows a linear correlation between CPU consumption and memory occupancy irrespectively of the forwarding algorithm. As expected, geographic forwarding grows more steeply than name-based forwarding (621 cycles per additional neighbour versus 71 cycles per additional FIB entry). Indeed, geographic forwarding requires the node to perform floating point multiplications to compute the distance to the next hops, while name-based forwarding only requires byte comparisons. It must be noted that we could implement geographic forwarding using fixed-point arithmetic or on a more recent CPU with embedded support for floating-point operations. However, while this would narrow the performance gap, geographic forwarding would still be more expensive in CPU cycles than the byte comparisons used by name-based forwarding.

While the gap between geographic and name-based forwarding appears to be important, it is just one component of the overall cost, which we detail in the next section.

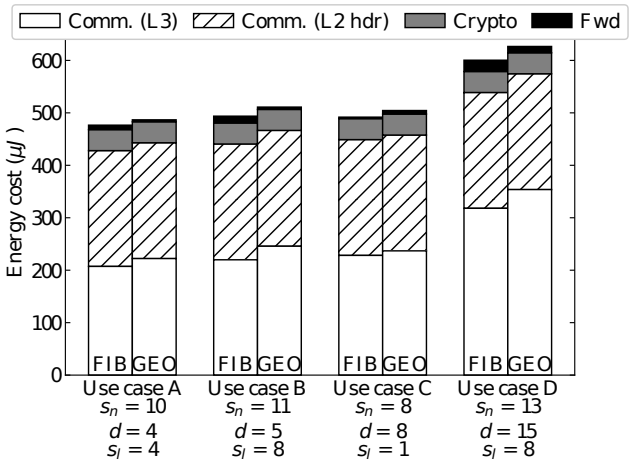


Figure 2.7 – Energy cost of the forwarding modules

2.5.4 Overall cost

We can now express the total cost for a node to relay an ICN packet as:

$$E_{relay}(algo, s) = E_{tx}(s) + E_{rx}(s) + 2E_{AES} + P_{CPU}\frac{n_{cycles(algo)}}{f_{CPU}} \tag{2.7}$$

It should be noted that since Data packets are forwarded through symmetric routing, they are not concerned by the computation overhead. Considering both the Interest and the Data packet, the forwarding cost per node adds up to:

$$E_{relay}(algo) = E_{relay}(algo, s_i) + E_{relay}(\text{exact-match}, s_c) \tag{2.8}$$

Finally, we can summarize $E_{forwarding}$ as the cost of forwarding an Interest packet and its corresponding Data packet over a multi-hop network by considering that on the path we have: (i) $T-1$ relay nodes that must perform both TX and RX operations, (ii) a source node that only performs TX for the Interest and RX for the Data packet, and (iii) a destination that performs only RX for the Interest and TX for the Data packet. This can be rewritten as T relay nodes performing both TX and RX operations for both packets:

$$E_{forwarding} = T(E_{relay}(algo, s_i) + E_{relay}(\text{exact-match}, s_c)) \tag{2.9}$$

In Figure 2.7, we represent the respective costs of each of the different components for the reference use cases in Table 2.1 with the corresponding numbers of FIB entries, neighbours and the size of the name. In particular, we group transmission and reception costs, and present them as L2 IEEE 802.15.4 and L3 ICN components, on which the cryptographic and forwarding costs are stacked. For each reference use case, we gather d and n_s from the literature, then compute $s_n = \log_2(n_s)$, and finally select a size of s_l relevant to the geographic scale of the use case (i.e., depending on the coordinate resolution).

From Figure 2.7, it clearly emerges that RX and TX operations are the predominant factors of energy consumption (even though they are actually underestimated as we do not account for MAC layer signalling nor idle listening [116]).

Table 2.7 – Additional bytes sent in the reference ICN-WSN architecture

Beacons (per node)	Interest packet (per packet & hop)
$58 + s_{id} + s_l$	$2 + s_l$

The communication cost is two orders of magnitude higher than the cost of forwarding (software), even for geographic forwarding with numerous neighbours, and one order of magnitude higher than the cost of cryptography (hardware). Notice that the cost of software forwarding lumps together energy expenditures related to CPU (selecting the forwarding face) and memory (storing and updating the neighbour table or the FIB), which are clearly negligible w.r.t. the energy spent on cryptographic and network operations.
Hence, to summarize:

— the principal overhead in energy consumption when using GPSR is the *increased header size* included in each Interest packet because of the GPSR TLV,

— the complexity of the forwarding algorithm is *clearly negligible* and Equation (2.7) becomes:

$$E_{relay}(algo, s) \approx E_{tx}(s) + E_{rx}(s) + 2E_{AES} \tag{2.10}$$

2.6 Cost of control traffic

An additional source of energy consumption is the background control information that is needed to discover routes to new names, new neighbours, and to maintain the network connectivity in spite of changes such as node mobility. That cost is intrinsically related to the geographic (Section 2.6.1) or flooding-based (Section 2.6.2) forwarding algorithm employed. In this section, we thus build a model to derive the energy consumption of the network forwarding process (data and control plane). We complete this model with simulation to estimate the spread of flooding for the F&L and MPR algorithms.

2.6.1 Geographic forwarding

For geographic forwarding, control information takes two forms: (i) the beacons to transmit geographic information between neighbours, and (ii) the additional GPSR TLV in the Interest packet to transmit per-packet forwarding state along the path. We now review the cost of both factors, which we summarize in Table 2.7.

Beacons. Beacons are the most obvious source of control overhead in geographic forwarding. As described in section 2.3.2, they are local (i.e., not routed) broadcast messages that do not propagate in the network and only reach the nodes involved in the L2 broadcast.

Let us consider a node whose immediate neighbourhood has changed: this node must broadcast its current position to its new neighbours, and receive the d broadcast messages from its neighbours. Given s_b the size of a beacon message,

the energy E_{change} required for this update by each node is simply:

$$E_{change}^{GPSR} = (s_b n_{tr,s} E_{tx}^b + E_{AES}) + d(s_b n_{tr,s} E_{rx}^b + E_{AES}) \tag{2.11}$$

Headers. On top of the beacons, additional control information for GPSR is embedded in every Interest packet through the TLV described in Section 2.3.3. As shown in Figure 2.4, this TLV contains both a flag for the forwarding mode (greedy or perimeter) and a set of coordinates. Thus, the size $s_{i,g}$ of an Interest packet for geographic forwarding is:

$$s_{i,g} = s_i + 2 + s_l \tag{2.12}$$

As we have seen in Figure 2.7, we expect the extra control fields in the headers to have a sizeable impact on data traffic, lowering the efficiency of GPSR with respect to name-based forwarding. We can now plug $s_{i,g}$ in Equation (2.9) and using Equation (2.10), we get:

$$E_{forwarding}^{GPSR} = T\left(4E_{AES} + n_{tr,s}(s_{i,g} + s_c)(E_{rx}^b + E_{tx}^b)\right) \tag{2.13}$$

At the same time, we expect the benefits of GPSR to come primarily from keeping the amount of FIB state bounded to the number of neighbours and limiting the exchanges at a local level, unlike F&L-based strategies.

2.6.2 Flood and learn

In the case of name-based forwarding, despite similar forwarding operations, the data-plane energy is lower due to the smaller Interest size ($s_i < s_{i,g}$):

$$E_{forwarding}^{F\&L} = T\left(4E_{AES} + n_{tr,s}(s_i + s_c)(E_{rx}^b + E_{tx}^b)\right) \tag{2.14}$$

However, we need to the estimate the cost of learning routes to objects in an F&L-based ICN-WSN deployment:

$$E_{change}^{F\&L} = \frac{N_{tr}(T,d)}{n_{tr,s}} 2E_{AES} + N_{tr}(T,d)(E_{rx}^b + E_{tx}^b)s_i \tag{2.15}$$

Therefore, we have to determine the number of network-wide transmissions required to update FIB information $N_{tr}(T,d)$, for which we model the propagation of a flooded packet over a wireless multi-hop network. Let us consider a uniform random geographic graph $G(N,d)$ where N is the number of nodes in G and d the average node degree, and let focus on a message m that must be flooded over the WSN with a maximum Time-To-Live (TTL) of T. Denoting with $N_{tr}(T,d)$ the number of times m has been transmitted during its propagation, including L2 retransmissions, we have:

$$\mathbf{E}[N_{tr}(d)] = \sum_{t=0}^{T-1} \mathbf{E}[n_{tr}^t(d)] \tag{2.16}$$

where $n_{tr}^t(d)$ is the number of times the message m is transmitted at the t-th hop (i.e., for a TTL$=T-t$). Letting V_t be the set of nodes that relay message

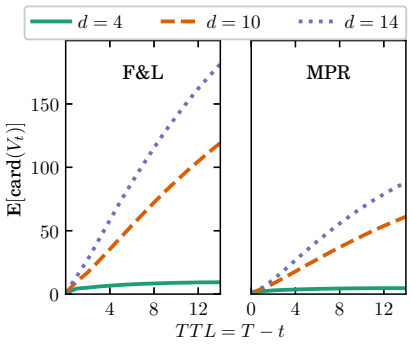


Figure 2.8 – Average value of $\text{card}(V_t)$

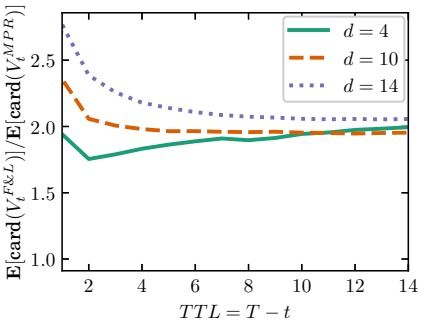


Figure 2.9 – Ratio between average values of $\text{card}(V_t)$ for naive F&L and MPR

m at hop t, and recalling that $n_{tr,s}(m)$ is the number of transmission attempts until a successful transmission, we have:

$$n_{tr}^{t}(d) = \sum_{i \in V_t} n_{tr,s}(p_c) \tag{2.17}$$

and thus:

$$\mathbf{E}[n_{tr}^{t}(d)] = \mathbf{E}[\text{card}(V_t)]\mathbf{E}[n_{tr,s}(p_c)] \tag{2.18}$$

To estimate the number of nodes $\text{card}(V_t)$ transmitting the message at the t-th hop, we simulate the packet propagation for both F&L and MPR. For this purpose, we have developed multi-threaded programs that we have made available to the community [133]. The tools generate random graphs with a given density and number of nodes and use a custom version of breadth-first search to compute $\text{card}(V_t)$ in the case of naive F&L. In the case of MPR, they compute the set of MPR-neighbours using the greedy algorithm described in [137]. For any given density, we sample a population of 10^5 random graphs on which we evaluate the number of transmissions for naive F&L and MPR from a random source. We run the simulations on a Linux 4.7 server with an Intel Xeon CPU clocked at 2.40GHz: for each density, the simulation takes about 9 hours, where the dominant [3] time is represented by the MPR strategy. As simulation time is rather long, we also provide on our GitHub the results of our simulation rounds as well as a Jupyter Notebook to explore them.

Figure 2.8 presents simulation results for the number of messages generated by naive F&L (left) and MPR (right) as a function of $T - t \in [0, 15]$ (x-axis) and for different density values $d \in [4, 15]$. To quantify the advantages brought by MPR, Figure 2.9 additionally reports the ratio of the messages generated by F&L over MPR. Interestingly, regardless of the density, MPR roughly halves the number of messages that need to be flooded at each step. To perform a conservative assessment of the benefits of geographic forwarding, we thus only consider MPR as a benchmark.

3. While Python is enough for F&L, we had to rewrite the tool in C for speed efficiency in the case of MPR. Both tools are available in the GitHub page

Finally, plugging in Equation (2.18) the $\text{card}(V_t)$ measured in simulations, we can accurately numerically estimate the forwarding cost of flood-based strategies:

$$\mathbf{E}[N_{tr}(d)] = \sum_{t=0}^{T-1} \mathbf{E}[\text{card}(V_t)] \frac{1 - p_c(d)^{M_{tr}}(M_{tr}+1) + p_c(d)^{M_{tr}+1} M_{tr}}{1 - p_c(d)} \quad (2.19)$$

where the collision probability $p_c(d)$ for IEEE 802.15.4 is given in [144] as a function of the average node degree (that is referred to as p_{netcol} in [144]). Results for Equation (2.19) are reported in Figure 2.10.

2.7 Guidelines for ICN-WSN operation

Using these results, we can systematically compare MPR and geographic forwarding for both energy (Section 2.7.1) and feasibility (Section 2.7.2) metrics. In particular, we apply the models derived in Section 2.5 and Section 2.6 to the four WSN reference deployments of Section 2.2, considering a network of OpenMote devices. Readers or ICN-WSN operators interested in other network characteristics are referred to our Jupyter Notebook [4], whose interactive interface can be used to explore more scenarios. For convenience, we summarize the main parameters in the evaluation in Table 2.6, as well as their default scalar or functional values when relevant.

2.7.1 Energy cost

Having modelled the energy budget required to transmit N_m exchanges, we can derive the number of Interest/Data exchanges for name-based forwarding ($N_m^{F\&L}$) and geographic forwarding (N_m^{GPSR}) that can be completed with the energy available in one AA battery ($E_{AA} = 15\,\text{kJ}$ [145]). For completeness, the message budgets are reported in Equation (2.20) and Equation (2.21).

4. https://github.com/marceleng/geographic-icthings/blob/master/models/geographic-icthings.ipynb

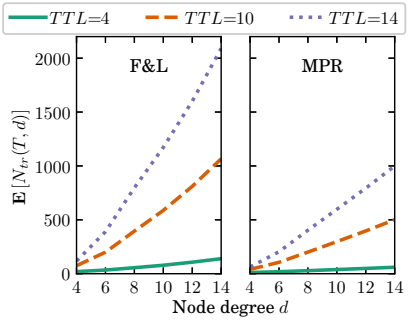


Figure 2.10 – Expected number of transmitted messages $\mathbf{E}[N_{tr}(T,d)]$

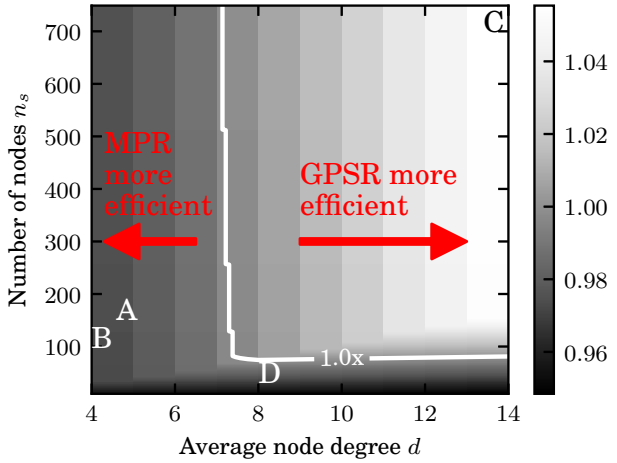


Figure 2.11 – Relative message budget $\frac{N_m^{GPSR}}{N_m^{MPR}}$ when $f_c = 60$

$$N_m^{MPR} = \frac{E_{AA} - \left(\frac{N_{tr}(T,d)}{n_{tr,s}} 2E_{AES} + N_{tr}(T,d) s_i (E_{rx}^b + E_{tx}^b)\right)}{\left(\frac{N_{tr}(T,d)}{n_{tr,s}} 2E_{AES} + N_{tr}(T,d) s_i (E_{rx}^b + E_{tx}^b)\right) / f_c + T\left(4E_{AES} + n_{tr,s}(s_i + s_c)(E_{rx}^b + E_{tx}^b)\right)} \tag{2.20}$$

$$N_m^{GPSR} = \frac{E_{AA} - \left((s_b n_{tr,s} E_{tx}^b + E_{AES}) + d(s_b n_{tr,s} E_{rx}^b + E_{AES})\right)}{\left((s_b n_{tr,s} E_{tx}^b + E_{AES}) + d(s_b n_{tr,s} E_{rx}^b + E_{AES})\right) / f_c + T\left(4E_{AES} + n_{tr,s}(s_{i,g} + s_c)(E_{rx}^b + E_{tx}^b)\right)} \tag{2.21}$$

These equations provide a metric for measuring the efficiency of both protocols: the number of possible Interest/Content exchanges (N_m) for a given battery capacity, depending on the density d, the average number of hops T and the topology change ratio f_c. Figure 2.11 shows the ratio between N_m^{GPSR}/N_m^{MPR} (i.e., the relative message budget of GPSR vs MPR) depending on the network size n_{nodes} (y-axis) and the average node degree d (x-axis). In this figure, the lighter the heatmap, the more efficient geographic forwarding is with respect to MPR. Without loss of generality, we select $T = 8$ and $f_c = 60$, which corresponds to the case where, if an Interest/context exchange happens every minute, then a route change happens every 1 hour.

Several conclusions can be drawn from Figure 2.11. First, let us note that the ratio takes values $N_m^{GPSR}/N_m^{MPR} \in [0.95, 1.05]$, i.e., the respective performances of name-based and geographic forwarding are close for these (T, f_c) settings. The inflexion point in the figure is caused by a change in the MPR regime: i.e., at some point, the network becomes big enough so that the flooding stops because of the TTL, rather than because everyone in the network already received the packet. Also, note that in these settings, 3 (out of 4) reference deployments (A, B, D) sit below the $N_m^{GPSR} = N_m^{MPR}$ contour and would thus (slightly) benefit from using MPR. Finally, it is easy to see that for an average node degree higher than 7 and a deployment size larger than 100 nodes, geographic forwarding performs (slightly) better than MPR.

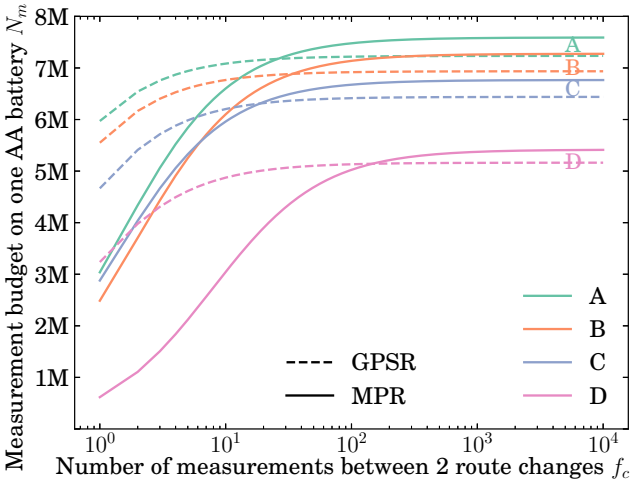


Figure 2.12 – Expected message budget for GPSR and MPR for the reference deployments as a function of network dynamism

We next turn our attention to dynamic cases where we vary the rate at which the network changes (e.g., due to node churn, mobility, deployment of new nodes, etc.). Specifically, f_c represents the number of consecutive data plane exchanges between two changes requiring control plane messages. We let $f_c \in [1, 10^4]$, so that for $f_c = 1$ there is a significant control plane overhead, whereas for $f_c = 10^4$ the network is mostly stable. We report the raw number of Interest/Data messages on an AA battery as a function of f_c for the four reference deployments on Figure 2.12. Two main observations hold. First, for networks with frequent changes geographic forwarding is up to twice as efficient as MPR for all considered scenarios. Second, it can be seen that MPR is slightly more efficient than geographic forwarding over relatively stable networks, although the difference is small enough not to have practical relevance. Thus, GPSR seems a good candidate for dynamic scenarios, while both GPSR and MPR can be used in stable topologies at approximately the same cost.

2.7.2 Memory and CPU complexity

Finally, we concisely summarize the CPU and memory footprint of geographic forwarding and F&L-based strategies in Figure 2.13. The contour plots show the relative footprint of GPSR versus F&L ICN, while the heatmap illuminates areas where GPSR is advantageous on both criteria (white), only in memory (grey) or in neither (black). The letters show the respective positions of the use cases described in Table 2.1. The picture shows that GPSR has a lower memory footprint when the number of FIB entries inflates, which is an important factor on memory-constrained nodes in networks where numerous names must be accessible. In particular, three of the use cases (A, B, and D) require less memory and CPU resources using GPSR rather than with name-based forwarding. Furthermore, while CPU consumption is often favourable to F&L ICN, GPSR is faster in sparse but large networks (e.g., $n_f > 40$ and $d < 5$).

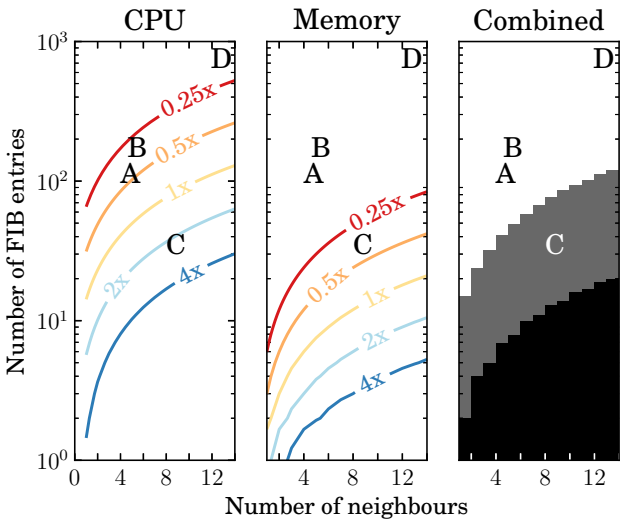


Figure 2.13 – Contours of the relative memory and CPU footprints of GPSR and F&L ICN. The heatmap shows areas where GPSR outperforms F&L ICN for both criteria (white), for memory utilisation (grey) or for neither (black)

Overall, there are no CPU/memory obstacles to implement geographic forwarding in ICN-WSN. Rather, GPSR can provide memory savings with respect to name-based forwarding, an appealing advantage for constrained environments.

2.8 Summary

In this chapter, we introduce a generic ICN architecture for WSNs, able of performing secure geographic forwarding. We use this framework to assess the CPU/memory feasibility of geographic forwarding as well as to derive energy models for geographic and name-based forwarding that encompass all network-related activities of a WSN. We implement GPSR in a RIOT-based ICN stack and contrast it to naive flooding, as well as to an improved flooding technique using Multi-Point Relay (MPR).

In summary, we find that GPSR-based forwarding is feasible in ICN-WSN. Specifically, GPSR memory requirements are lower than that of flood-based strategies. Additionally, while algorithmic complexity increases when using GPSR over flooding-based strategies, the required CPU resources are a negligible component of the overall energy cost. Indeed, the cost of security (including cryptographic operations and network overhead) is at least one order of magnitude higher than the computation cost of the forwarding algorithm, which can, therefore, be neglected.

In terms of energy consumption, two opposite forces are in play: the increased header size in the case of GPSR translates into higher energy cost per unit packet while GPSR keeps beaconing local, lowering the cost of network-wide updates. As such, while there is a clear incentive in using GPSR for dynamic networks requiring frequent updates, the performance gap in the case of networks with stable topologies and infrequent changes is slightly favourable to

flood-based strategies. At the same time, that gap remains small and it should not be a limiting factor for the use of GPSR in ICN-enabled WSNs.

Chapter 3

Enforcing latency-control in the Fog via popularity-aware admission control

While sensor networks are not only useful for generating data but can implement primary levels of environment automation, many IoT applications require consequent computing or storage resources and thus cannot be implemented on low-power nodes such as the ones described in Chapter 2. Complementary to the study of routing in WSNs is thus the study of routing and forwarding data extracted from the WSNs towards the corresponding applications. In that regard, as discussed in Section 1.1.2, two complementary platforms have emerged: the *Cloud*, located in a large data centre designed for scale, and the *Fog*, located at the edge of the network for latency-constrained applications. In this chapter, we consider the problem of controlling admission to a Fog platform (and offloading the refused traffic to the Cloud) so as to guarantee an average response time. In particular, exploiting the request-awareness provided by ICN, we look at how network-level popularity estimation can be used to realize line-rate, low-latency, efficient admission control for a Fog deployment.

3.1 Admission control for QoS in Fog deployments

The emergence of low-latency applications in the IoT has created a need for computing and storage platforms located geographically and topologically close to the access. Introduced as an extension of the Cloud at the network edge for computing and storage purposes [17], the Fog is a highly virtualized, potentially distributed, computing and storage platform placed at the edge of the network and capable of processing IoT data under low latency. Since its definition, there has been a growing interest in the research community for Fog computing, encompassing areas such as workload placement [146], caching [48, 71], or application profiling [147]. Accelerated by the advent of 5G, the Fog is expected to play a key role for IoT applications in cases where latency requirements and/or security reasons preclude the use of a Cloud-only approach.

Fog platforms are not intended as a replacement for Cloud processing, rather

as a complementary computing and storage platform to use if and whenever beneficial. For instance, with frameworks such as AWS Greengrass [148], IoT providers can use their own devices (e.g., an IoT gateway or a local compute node) to perform some stateless pre-processing of data on its path to the Cloud. In addition, the Fog does not enjoy the same elasticity as the Cloud. Indeed, while Cloud data centers inherently scale thanks to their size and their high number of tenants, Fog nodes have strong physical limits that cannot be infringed. Thus, sudden bursts of requests, e.g., flash crowds, can greatly increase application response time and raise scalability issues for Fog deployments. In such cases, the natural solution is to redirect part of the Fog load to the Cloud [149, 150]. However, if done incorrectly, that could actually worsen the response latency. Furthermore, renting compute and storage power in the Cloud is expensive, as opposed to user-owned Fog devices. In this chapter, we thus investigate the following question: *how to minimize Cloud costs while offering statistical latency guarantees in case of sudden bursts of requests in a Fog network?*

As described in Section 1.1.3, IoT applications can roughly be categorized in three QoS categories: (i) *latency-critical*, where processed data must be received within 1-10 ms, (ii) *latency-sensitive*, where the expected response time is of the order of 100 ms [34], and (iii) *latency-tolerant*, that have no specific delay constraint. Whereas latency-critical applications cannot run in the Cloud (but rather in the device or at Fog level), latency-tolerant applications would naturally be deployed there. Thus, Cloud redirection is most relevant for applications of the latency-sensitive class, where the computing bottleneck in the Fog may force to offload part of request processing to the Cloud. At the same time, the use of faraway Cloud resources not only increases the *cost* for the operator but may also increase the *service latency*. There is thus a need for clever Fog admission control (AC) to keep response time bounded.

In this chapter, an approach for Fog admission control is presented that relies on using request popularity to optimize the usage of the Fog platform, and in particular of the caching capacities of the Fog. Specifically, two AC strategies are introduced: the LFU-AC, which exploits *perfect knowledge* of the request popularity distribution and the LRU-AC, which is based on statistically estimating the most popular requests by using a cache on request identifiers with the Least-Recently-Used (LRU) policy. Using an analytical model, the LFU-AC is shown to provide sizeable benefits over optimized request-agnostic AC strategies. The LRU-AC performance is also shown to be close to the LFU-AC bound.

A system implementation of the LRU-AC strategy on programmable hardware is then proposed, thus allowing for a line-rate, low-latency deployment that does not require provisioning extra computing resources. As implementing an LRU cache in hardware is inconvenient due to the necessity of using linked-list structures [151], an implementation of the LRU-AC using Ageing Bloom Filters (ABF) [152] in NetFPGA [153] is proposed. An analytical model of the expected hit-rate of the ABF is built to guarantee its behaviour w.r.t. the LRU-AC. The use of an ABF is shown to reduce memory footprint by up to 4, making the implementation fit within memory sizes available in programmable hardware devices. The implementation relies on hICN [154], an ICN protocol that uses the semantic of IP/TCP for backward compatibility. Using the P4 framework [155], request semantic is then efficiently accessible in hardware at the network layer thanks to hICN's *name-based* forwarding with *fixed-sized* names, while relieving

the AC from maintaining per-flow state thanks to a *connection-less* approach. The implementation is shown to support a throughput of 16.7 Mpps (exceeding 10 GbE line-rate) with a packet processing latency of 3 µs. Algorithmic performance of the ABF is evaluated and shown to produce QoS results that are consistent with the LRU-AC model.

This chapter has been the object of two publications. [100] introduces the LRU- and LFU-AC strategies and provides a preliminary evaluation based on the queueing model. [97] describes the system implementations of the LRU-AC using ABFs. Note that, while orthogonal to this work (and thus not reported in this thesis), [99] also exploits the LRU-AC strategy to optimize cache hit in the context of video delivery networks.

The remainder of this chapter is organized as follows: Section 3.2 states the Fog-AC problem in a principled way; Section 3.3 introduces the analytical model used to derive the performance of AC strategies; relevant popularity-based strategies are presented and partially evaluated in Section 3.4; in Section 3.5, the use of ABF to realize the LRU-AC, selected because it is both practical and effective, is proposed and justified; the LRU-AC hardware implementation is then introduced in Section 3.6 and evaluated in Section 3.7.

3.2 Problem description

3.2.1 Reference Fog architecture

The reference IoT architecture assumed in this chapter is a simplification of the larger picture depicted in Section 1.1.2 and consists of three main components: (i) IoT networks, where sensors, actuators, and users are connected; (ii) an access network which connects these IoT networks together and with the Internet; and (iii) a Cloud platform, used for compute and storage. On top of this Cloud platform, a Fog deployment is available in the access network. Both the Fog and the Cloud are equipped with LRU caches. Without loss of generality, we adopt a per-application view and consider a single Fog node with computing and storage capabilities that receives homogeneous requests from a single stateless application (e.g., lambda function). The Fog, either owned by the application developer or rented to its Internet Service Provider (ISP), is deployed for latency-critical applications. However, as Fog platforms are not elastic, the application developer wants to use the residual computing and memory capacities to perform latency-sensitive tasks. That residual capacity is considered to be constant, hence the Fog node cannot handle a high arrival rate for a prolonged period. It redirects part of its load to a Cloud platform, where the operator rents computing and storage resources, while still respecting the latency agreement.

An admission control module is in charge of forwarding incoming requests from the IoT networks to either Fog (accept) or Cloud (refuse). That architecture is summarized in Figure 3.1. We consider Fog applications working in the following way: (i) the application retrieves *raw data* from one or several sensor nodes (e.g., an image or a temperature from several sensors); (ii) it performs some computation to transform the raw data into *processed data* (e.g., JSON file indicating detected shapes, or the average of the measured temperatures); (iii) the processed data is retrieved by users or actuators which use it to take decisions. As security is paramount for Fog applications [17], both processed and

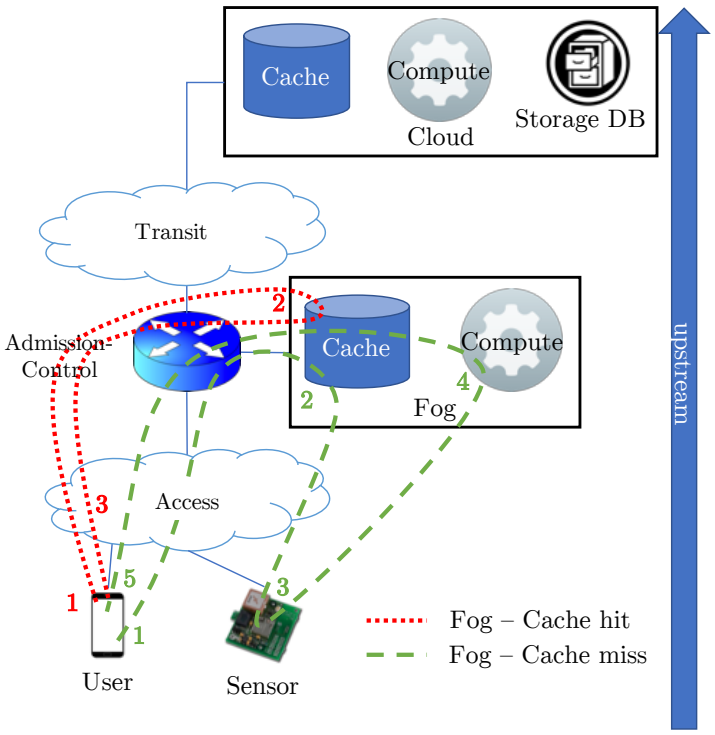


Figure 3.1 – Reference IoT, Fog and Cloud architecture

raw data are encrypted during network transmissions, commonly with (D)TLS. In particular, we consider a pull-based model driven by client requests, of which we illustrate two of the possible paths in Figure 3.1. The user application starts by issuing requests (step 1), which reach the admission control module. For the sake of illustration, only cases where requests are accepted in the Fog are shown. In the Fog node, the request is matched against a cache (step 2) for the availability of processed data. In case of a *cache hit* (red dots ···), the processed data is sent back directly to the user. In case of a *cache miss* (green dashes ---), the raw data must be retrieved from the sensor (step 3), before the computation can take place (step 4) and the processed data can then be served back to the user (step 5).

3.2.2 Fog vs Cloud admission control

To devise such strategies where the application runs both in the Cloud and in the Fog, costs for the Fog operator must be considered. In the Cloud, resources are elastic and the capacity can be increased as the incoming load grows [156]. Furthermore, the Cloud is assumed to always store the raw data for orthogonal archiving and monitoring purposes; the cost of raw data archival is thus not considered. This comes on top of the cache for processed data, whose size is defined by the amount of storage rented in the Cloud. Moving data to/from the Cloud through the transit network also has a cost. On the other hand, application deployment in the Fog comes at no cost for the Fog operator (since it owns the infrastructure). As Fog nodes have limited storage resources, the Fog

cache is only used for processed data. The Fog node also has a finite amount of computing resources, which must be equally shared between all incoming requests. Thus, in case of a high load, the Fog node might have a high completion time or even start dropping requests, which may violate the agreement set up with the Fog application developer.

The need for a proper offloading function ϕ between the Fog and Cloud resources thus becomes clear: such a function should *minimize the cost* $\Pi(\phi)$ of renting Cloud resources while *respecting the agreed upon latency constraint*, which we outline below (and formalize in Section 3.3):

$$\begin{cases} \text{min.} & \Pi(\phi) \\ \text{s.t.} & \mathbf{E}[T(\phi)] \leq \Delta \end{cases} \tag{3.1}$$

where T is the stochastic variable describing the system completion time. In particular, Π is a cost per second, in accordance with the pay-as-you-use model for Cloud pricing. We use a *statistical* latency constraint, which guarantees that the average request completion time is under Δ. The advantage of such formulation is clear considering that it enables us to express the constraint in closed form in a queueing model, which simplifies tractability.

3.3 An analytical model

To understand the performance of a given admission control function, a queueing model that describes the systemic behaviour of the IoT architecture is introduced. The variables necessary for the model are summarized in Table 3.1.

3.3.1 Application model and request distribution

Let us consider a single application running on a sliced Fog deployment. This application is described by its latency constraint Δ, its raw data size s_r, its processed data size s_p, and its amount of service work X. In particular, we assume that s_r and s_p are constant, while X is a stochastic variable following an exponential distribution.

Let now R be the total number of possible requests, and $\{r_1, \ldots, r_R\}$ these requests. Following previous work, we consider that the request popularity distribution q follows a Zipf distribution [48, 157–159], i.e., for a request r arriving in the system, $q(k) = \mathbf{P}[r = r_k] = \gamma k^{-\alpha}$ where $\alpha > 0$ is the skew parameter and γ a normalization factor. In particular, requests arrivals are user-driven and are thus well modelled by a Poisson process of parameter λ. The arrival processes for each request r_k are assumed to be independent and thus follow a Poisson process of parameter $\lambda_k = q(k)\lambda$. The use of an independent requests model (IRM) is known to under-estimate of the cache benefits [160, 161], so we expect our results to be conservative.

3.3.2 Queueing model

Whenever relevant, we follow seminal work [149, 162–164] to select the most appropriate queue to describe each resource. Particularly, an M/M/1-PS queue is selected for the *Fog compute* [149, 162]: processor-sharing policy models that

Table 3.1 – Variables used in the model

Application specific		**unit**
Number of possible requests	R	-
Cumulated arrival rate	λ	req/s
Amount of service work	X	CPU cycles
Raw data size	s_r	B
Processed data size	s_p	B
Application latency constraint	Δ	s
Request popularity distribution	$q(r)$	-
Optimization variables		
Load-balancing function	$\phi(r)$	-
Cloud cache size	s_c	Objects
Total request serving time	$T(\phi, s_c)$	s
Cost function	$\Pi(\phi, s_c)$	\$/s
Fog characteristics		
Fog compute capacity	p_f	Hz
Fog cache size	s_f	Objects
Access network capacity	κ_a	B/s
Access network propagation time	τ_a	s
TLS establishment delay	τ_f	s
Cloud characteristics		
Cloud compute capacity	p_c	Hz
DB query delay	τ_d	s
Transit network capacity	κ_t	B/s
Transit network propagation time	τ_t	s
TLS establishment delay	τ_c	s
Cloud pricing		
Compute price	c_c	\$/s
Network price	c_n	\$/s
Storage price	c_s	\$/B
Miscellaneous		
Cache hit probability	$h_{\substack{f \\ c}}(r, s_{\substack{f \\ c}})$	-

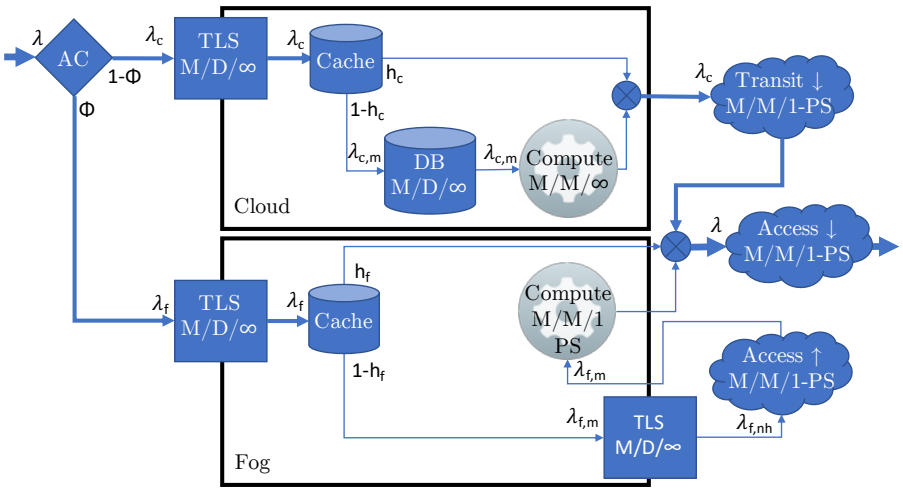


Figure 3.2 – Queueing network

the Fog has a fixed amount of resources that must be shared between all the incoming requests. On the other hand, as the *Cloud compute* is elastic, it is best represented by an $M/M/\infty$ queue [1], and we represent *Cloud-database* access as a constant-time $M/D/\infty$ queue. For *network resources*, the $M/M/1$-PS model is used as common in the literature [149, 163, 164]. As *admission control* decisions and cache lookup should be done at line-rate (see Section 3.6), their impact is considered minimal w.r.t. other queues; and since they do not impact the comparison between Cloud and Fog service time anyway, they are neglected in what follows. Finally, the TLS endpoints are modelled as $M/D/\infty$ queues, neglecting the computation time of the TLS handshake. Assuming that TLS is running in version 1.3, only 1 round-trip is necessary to establish the TLS connection, i.e.:

$$\tau_f = 2\tau_a$$
$$\tau_c = 2(\tau_a + \tau_t).$$

The resulting queueing system is depicted in Figure 3.2. One can note that the request transmission time is uniquely taken into account in the TLS queue: since IoT requests have negligible size, their transmission time is indeed dominated by their propagation time.

The AC strategy can be expressed as $\phi{:}\{1, \ldots, R\}{\to}[0, 1]$, a function that to each request associates a probability of being accepted in the Fog. In particular, given a popularity distribution q and an admission control strategy ϕ, the popularity distribution of requests arriving in the Fog is $q_f(r) = \gamma_f \phi(r) q(r)$, where γ_f is a normalization factor. For computing the hit probability in the Fog cache, the seminal approximation proposed by Che et al. [165] is used:

$$h_f(r) \approx 1 - e^{-q_f(r) t_s} \tag{3.2}$$

where t_s is the unique root of $\sum_{r=1}^{R}(1 - e^{-q_f(r)t}) = s_f$. A similar model applies

1. The underlying assumption is that of a perfect autoscaling policy.

Table 3.2 – Arrival rate per queue in network

AC	AC module	λ
	Access down.	
Fog	TLS - Fog	$\lambda_f = \sum_{r \in R} \phi(r) \lambda q(r)$
	Cache - Fog	
	Access up.	$\lambda_{f,m} = \sum_{r \in R} \phi(r)(1 - h_f(r)) \lambda q(r)$
	Compute - Fog	
Cloud	TLS - Cloud	$\lambda_c = \sum_{r \in R} (1 - \phi(r)) \lambda q(r)$
	Cache - Cloud	
	Transit down.	
	DB	$\lambda_{c,m} = \sum_{r \in R} (1 - \phi(r))(1 - h_c(r)) \lambda q(r)$
	Compute - Cloud	

for the Cloud cache, replacing the probability function ϕ by its complement $1 - \phi$.

3.3.3 Computing the statistical latency

First, let us point out that since processor-sharing queues are quasi-reversible processes, the exit distribution of an M/G/1-PS queue is a Poisson process (Theorem 3.6 of [166]). This is also true for the M/G/∞ queue [167], justifying that all the queues have a Markovian input. The expected sojourn time in each of the queues is then easy to compute. In particular, the expected service time for requests with an amount of service work X and Poisson arrival rate λ in an M/G/1-PS of capacity C is given by:

$$\mathbf{E}[T] = \frac{1}{(\mu - \lambda)} \quad \text{where } \mu = \frac{C}{\mathbf{E}[X]} \tag{3.3}$$

The arrival rate at each queue can be derived from the offloading strategy ϕ and the cache hit probabilities h_f and h_c at the Fog and Cloud caches respectively, obtained using Equation (3.2). Per-queue arrival-rates are reported in Table 3.2.

We can thus get the equation for the expected queueing delay as:

$$\mathbf{E}[T] = \sum_r q(r) \Big[\phi(r) \mathbf{E}[T_f(r)] + (1 - \phi(r)) \mathbf{E}[T_c(r)] + \mathbf{E}[T_{a,d}] \Big] \tag{3.4}$$

where the expected latency for the service time in the Fog $T_f(r)$ is:

$$\mathbf{E}[T_f] = \tau_f + (1 - h_f) \left(\tau_f + \mathbf{E}[T_{a,u}] + \mathbf{E}[T_{comp,f}] \right),$$

the expected latency for the service time in the Cloud $T_c(r)$ is:

$$\mathbf{E}[T_c] = \tau_c + (1 - h_c) \left(\tau_d + \frac{\mathbf{E}[X]}{p_c} \right) + \mathbf{E}[T_{transit,d}],$$

and where $T_{a,u}$, $T_{a,d}$, $T_{comp,f}$, $T_{transit,d}$ respectively represent the time spent in the access upstream, in the access downstream, in the compute in the Fog, and in the transit downstream, and whose expected values can be computed with Equation (3.3), Table 3.1 and Table 3.2.

3.3.4 Computing the cost function

The cost per second consists of a network, a computing, and a storage term. The computing power rented in the Cloud is assumed to be synchronized with the incoming load (i.e., the Cloud spawns a container process at each new request). The cost of running the Cloud increases proportionally to the requested load: $p(c, s, n) = c_c c + c_s s + c_n n$, where c (resp. s) is the amount of computing (resp. storage) resources rented on the Cloud, and ν is the egress Cloud traffic in bytes per second.

Compute cost

Since the resource consumption in the Cloud is assumed to be elastic, if $Q_c(t)$ is the number of customers in the Cloud compute M/M/∞ queue, the instantaneous number of instantiated Cloud computing instances is: $c(\phi, s_c)_t = Q_c(t)$. According to [167], the expected value for $c(\phi, s_c)$ is thus:

$$\mathbf{E}[c(\phi, s_c)] = \frac{\lambda_{c,m}(\phi, s_c)}{p_c/\mathbf{E}[X]}$$

Storage cost

The storage cost depends on the cache size in the Cloud: $s(\phi, s_c) = s_c s_p$.

Network cost

For each incoming request, s_p is transferred downstream as a reply. Given the Cloud arrival rate λ_c, the average number of bytes per second on the Cloud downstream link is:

$$\mathbf{E}[\nu(\phi, s_c)] = \lambda_c(\phi) s_p$$

Thus, the total cost function reads:

$$\Pi(\phi, s_c) = \frac{c_c \lambda_{c,m}(\phi, s_c)}{p_c/\mathbf{E}[X]} + c_s s_c s_p + c_n \lambda_c(\phi) s_p \tag{3.5}$$

3.3.5 An example application - Numerical parameters

All upcoming numerical evaluations use the characteristics described in Table 3.3. In particular, we select an application with a medium computing difficulty (10 ms on a 1 GHz processor) and medium processed data size. For the Fog deployment, the application is assigned a slice of a computing platform amounting to a 3 GHz CPU and 1 GB of cache. Finally, to make the evaluation more realistic, the public pricing of the Google Compute infrastructure as of October 2017 is used, the delay target is set to Δ=100 ms.

3.4 Popularity-based Fog admission

3.4.1 Optimizing Fog resources

Request processing in the Fog-Cloud system can be decomposed in three modes: requests served (i) from the Fog cache, (ii) from the Fog compute, and

Table 3.3 – An example application

Deployment		Application	
p_f	3 GHz	R	10^7
s_f	10^5 (1 GB)	λ	10 kHz
κ_a	10 Gbit/s	$\mathbf{E}[X]$	10^7 CPU cycles
τ_a	2 ms	s_r	1 MB
p_c	2 GHz	s_p	10 kB
τ_d	1 ms	α	1
κ_t	1 Gbit/s	Δ	100 ms
τ_t	20 ms		

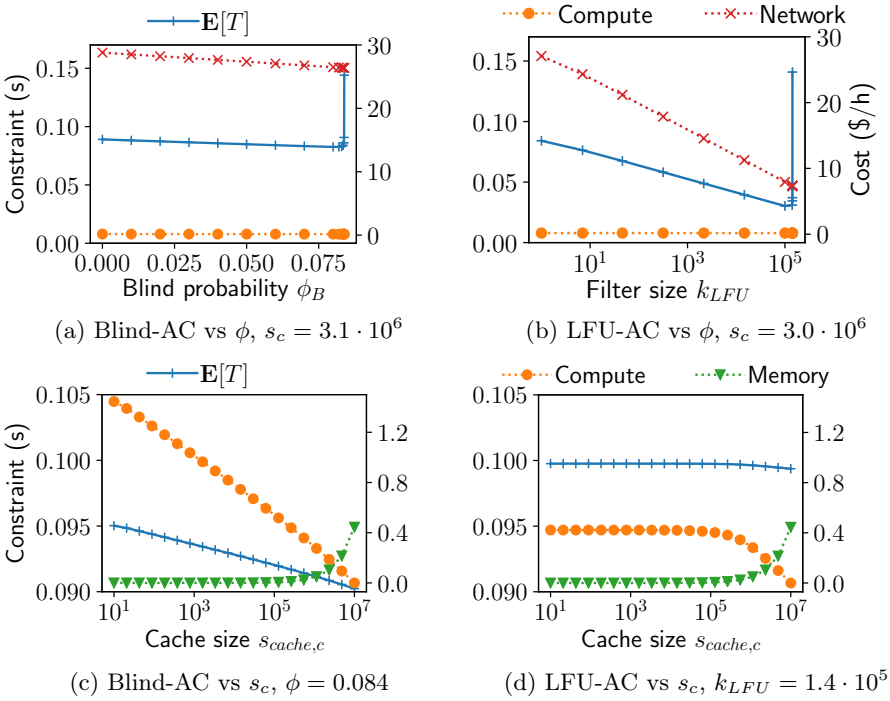
Cloud pricing	
c_n	\$0.08 per GB
c_s	\$0.004446 per GB and hour
c_c	\$0.033174 per vCPU and hour

(iii) from the Cloud (disregarding the Cloud cache vs the Cloud compute trade-off). An effective AC strategy should then maximize the proportion of traffic handled by the Fog. Since the Fog compute has a fixed capacity, increasing the amount of traffic handled by the Fog can only be done by increasing the cache hit-rate. The AC policy should then aim at maximizing the global Fog cache hit-rate (as defined by $\widetilde{h_f} = \sum_r \phi(r) h_r(r, \phi)$) while keeping the Fog-Compute arrival rate $\lambda_{f,m}$ bounded.

To this end, an effective approach is to admit only popular content in the Fog. Indeed, this method increases the hit-rate of the cache policy effectiveness by artificially reinforcing the skewness towards the most popular pieces of content in the cache input distribution. The admission control must then be able to (i) identify the content targeted by each request and (ii) extract popularity patterns. Content identification (i) is an architectural problem, which we discuss in Section 3.6.1. To illustrate the limit of solutions without content awareness, a "*blind*" admission control is introduced in Section 3.4.2. Extracting popularity patterns (ii) is a similar problem to caching policies: detecting the most popular pieces of content. Thus, using a virtual cache (caching only *identifiers* rather than content) that follows traditional admission and eviction policies is a natural solution. In this case, a hit in the virtual cache identifies popular requests which should be handled in the Fog, while a miss hints that the request is not popular and should be offloaded to the Cloud. This solution is similar to multi-layered caching systems such as the 2Q-LRU [168], differing mainly in that the first layer (the identifier cache) is completely independent from the actual cache. Indeed, in 2Q the virtual layer only governs cache insertion (but not cache retrieval), whereas the Fog-AC might refuse requests even though the corresponding answer is available in the Fog cache. In particular, the LFU (Section 3.4.3) and LRU (Section 3.4.4) policies are investigated.

3.4.2 Blind admission control

As a baseline, a request-oblivious admission control strategy, called *Blind-AC*, is introduced. It blindly load-balances all traffic with i.i.d. probability:



(a) Blind-AC vs ϕ, $s_c = 3.1 \cdot 10^6$

(b) LFU-AC vs ϕ, $s_c = 3.0 \cdot 10^6$

(c) Blind-AC vs s_c, $\phi = 0.084$

(d) LFU-AC vs s_c, $k_{LFU} = 1.4 \cdot 10^5$

Figure 3.3 – Constraint value (left) and cost (right) of the system vs ϕ and s_c for the Blind- and LFU-AC at fixed cache size

$\phi(r) = \phi_B, \forall r$. In particular, Equation (3.1) can be rewritten as:

$$\begin{cases} \text{min.} & \Pi(\vec{\phi}_B, s_c) \\ \text{s.t.} & \mathbf{E}[T(\vec{\phi}_B, s_c)] \leq \Delta \end{cases}$$

with $\vec{\phi}_B = (\phi_B, \ldots, \phi_B)$. In this particular case, since $(\phi_B, s_c) \in [0,1][0,R]$, the problem is easy to optimally solve numerically.

We next evaluate the respective importance of the two optimization variables using the parameters reported in Table 3.3 for numerical evaluation. In particular, the variation of the costs and constraint functions depending on either ϕ_B or s_c are represented in respectively Figure 3.3a and Figure 3.3c.

Figure 3.3a shows the variation of the constraint function (+, left side) and of the compute (•, right side) and network (, right side) costs for a fixed cache size $s_c = 3.1 \cdot 10^6$. The storage cost is ignored as it is constant (since s_c is fixed). The first takeaway is that, as expected from the costs in Table 3.3, the network cost is dominant w.r.t. the compute and memory cost. We also notice that the constraint function diverges towards $+\infty$ when ϕ_B grows close to 0.085, as the Fog compute queue becomes unstable and cannot handle the request rate.

In Figure 3.3c, we next vary the cache size s_c for a fixed load-balancing probability $\phi_B = 0.084$ (the sweet spot in Figure 3.3a). We show the value of the constraint function (+, left side), and the compute (•, right side) and memory (▼, right side) costs. The network cost is now ignored since it is constant when ϕ is constant. First, we note that varying the cache size has a limited impact on

both the cost and the constraint function. Furthermore, in this case, given that the compute is almost one order of magnitude more expensive than the memory and the Cloud popularity distribution is sufficiently skewed, it is interesting to cache highly popular requests.

3.4.3 LFU-AC strategy

Let us first consider a perfect LFU virtual cache, which deterministically identifies the k_{LFU} most popular requests for processing in the Fog. In particular, the admission control function ϕ can be expressed as:

$$\phi(r) = \delta_{r \leq k_{LFU}} \coloneqq \begin{cases} 1 & \text{if } r \leq k_{LFU} \\ 0 & \text{otherwise.} \end{cases}$$

Equation (3.1) then becomes:

$$\begin{cases} \text{min. } \Pi(\vec{\delta}_{k_{LFU}}, s_c) \\ \text{ s.t. } \mathbf{E}[T(\vec{\delta}_{k_{LFU}}, s_c)] \leq \Delta \end{cases}$$

with $\vec{\delta}_{k_{LFU}} = (\delta_{1 \leq k_{LFU}}, \ldots, \delta_{R \leq k_{LFU}})$. Again, this problem is two-dimensional and can easily be solved numerically.

Figure 3.3b (resp. Figure 3.3d) shows the evolution of the cost and constraint functions while setting $s_c = 3.0 \cdot 10^6$ (resp. $k_{LFU} = 1.4 \cdot 10^5$) for the setup in Table 3.3. At a first glance, Figure 3.3b indicates that a proper choice of k_{LFU} allows decreasing the network cost at levels unreachable with the Blind-AC while respecting the constraint. Furthermore, as in Section 3.4.2, the dominant factor in terms of cost is the number of offloaded requests. Both of these insights point towards LFU as a good strategy for Fog/Cloud admission control. Additionally, Figure 3.3d shows that for small values of s_c, the compute cost stays constant. This is due to the popularity distribution at the Cloud cache, which only contains the long tail of the Zipf distribution. Thus, for small cache sizes, the hit probability is low and the cache almost useless.

3.4.4 The LRU-AC strategy

While the LFU-AC is efficient, it is an ideal policy, difficult to realize in practice if the popularity distribution is not known in advance (as deriving the exact popularity distribution is difficult and slow [2]). To derive a practical admission control policy, we argue that the admission control does not need to learn the popularity of each specific request. It only needs to flag whether a request is popular, acting as a low-pass filter. In a second step, the LRU policy is thus considered for the admission control virtual-cache. The functioning of the corresponding admission control module, called the LRU-AC, is summarized in Figure 3.4. Compared to the aforementioned counter solution for the LFU-AC, the LRU-AC has four main advantages: (i) it does not require prior knowledge of the application; (ii) it keeps memory constrained to the size of the filter, instead of the size of the catalogue; (iii) it is flexible w.r.t. changes in the popularity

2. This requires either offline analysis of the popularity distribution, or to keep counters of incoming requests. Both solutions are not flexible to popularity changes and are difficult to implement efficiently.

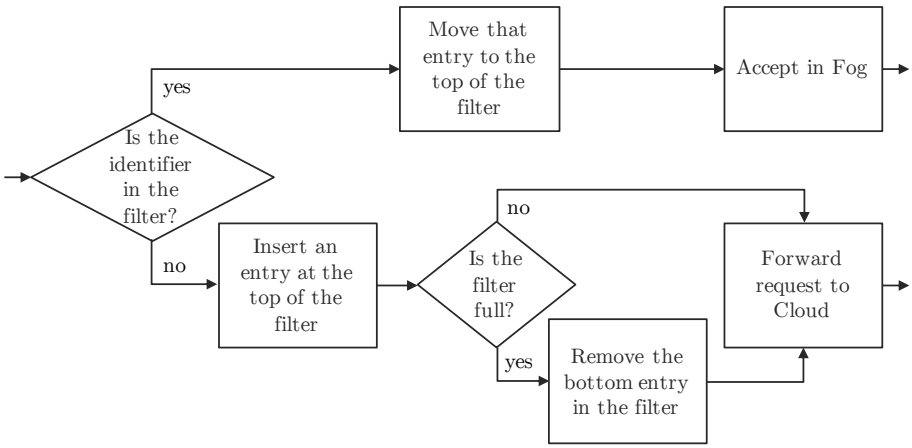


Figure 3.4 – Functionning of the LRU-AC

distribution; (iv) it requires minimal effort for integration in ICN forwarders as the LRU structure is already used for caches.

To incorporate LRU-AC in the model, we must compute the load-balancing function ϕ depending on the filter size k_{LRU}. Since the admission control behaves like an LRU cache, $\phi(r) = h_{k_{LRU}}(r)$ where $h_{k_{LRU}}(r)$ is the hit probability for the request r in an LRU cache of size k_{LRU} with input distribution q, which can be derived straightforwardly from Equation (3.2). Integrating this in Equation (3.1) yields a constraint and a cost function that depends only on k_{LRU} and s_c:

$$\begin{cases} \text{min. } \Pi(\vec{h}_{k_{LRU}}, s_c) \\ \text{ s.t. } \mathbf{E}[T(\vec{h}_{k_{LRU}}, s_c)] \leq \Delta \end{cases}$$

with $\vec{h}_{k_{LRU}} = (h_{k_{LRU}}(1), \ldots, h_{k_{LRU}}(R))$. However, the interaction between the LRU meta-cache and Fog LRU cache introduces some correlation effects, as shown by Garetto et al., for the LRU-2Q cache [169]. Following their example, a discrete time Markov chain is used to model the interdependence between hits in the filter and hits in the Fog cache. Details of the derivation are provided in Appendix A.1. Compared to the LFU-AC, realizing and optimizing the LRU-AC only requires knowing the popularity skewing factor α and the arrival rate λ instead of the actual per-content popularity distribution.

3.4.5 Preliminary evaluation of the admission control strategies

In this section, a first evaluation of the admission control strategies based on the model introduced in Section 3.3 provided. In particular, the Method-of-moving-asymptotes [170] (in its NLopt implementation [171]) is used to solve the optimization problem (Equation (3.1)). A Jupyter notebook is available for reproducing the results of this section or to experiment with different parameters [172].

In a first step, we show in Table 3.4 the optimized values for our example application. We first note that using the LRU and LFU-AC allows the Fog to handle more than twice as many requests as with the Blind-AC. This results

Table 3.4 – Optimal costs and parameters per AC

Method	$\mathbf{E}[\phi]$	s_c	Π
Blind	0.084	$3.1 \cdot 10^6$	27 \$/h
LFU	0.75 ($k_{LFU} = 1.4 \cdot 10^5$)	$3.0 \cdot 10^6$	7.6 \$/h
LRU	0.71 ($k_{LRU} = 2.8 \cdot 10^5$)	$2.9 \cdot 10^6$	8.6 \$/h

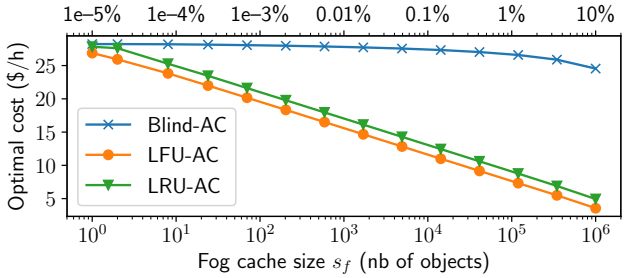


Figure 3.5 – Cost vs size of Fog cache for Blind-, LFU- and LRU-AC

in a decrease in offload cost of more than 70%. Furthermore, it shows that the LRU-AC has similar performances to the LFU-AC, with a 3% relative difference in offload cost w.r.t. the Blind-AC.

In Figure 3.5, the influence of the Fog cache size s_f on the optimal Cloud cost for the Blind-, LFU-, and LRU-AC strategies is shown. In particular, this figure shows that both the LFU- and LRU-AC strategies are much better at exploiting an increasing cache size that the Blind-AC. Indeed, for the Blind-AC, increasing the Fog cache size only slightly decreases the optimal cost. On the other hand, both the LFU- and the LRU-AC provide an exponential decrease of the cost as the cache size increases, showing the effectiveness of popularity-based admission control. Furthermore, this graph confirms that the LRU-AC is an extremely good approximation of the ideal LFU-AC, and regardless of the Fog cache size.

Figure 3.6 then investigates the influence of the popularity distribution skew (represented by the Zipf parameter α). For small α values, the popularity distribution converges towards a uniform distribution, thus diminishing the impact of popularity-based ACs. However, for typical values of α found in the literature ($\alpha \in [0.5, 1.1]$), the LFU- and LRU-AC strategies allow for a largely reduced optimal cost for both Fog cache sizes that we tested. Furthermore, the LFU- and LRU-AC strategies with $s_f = 10^4$ are much more efficient than the Blind-AC strategy with $s_f = 10^5$. This indicates that the strategies also allow for more efficient provisioning of Fog resources. Once again, the performance of the LRU-AC is close to the LFU-AC, varying by at most 6%.

Finally, the impact of the arrival rate on the efficiency of the strategies is shown in Figure 3.7. Interestingly enough, the optimal cost increases linearly w.r.t. the arrival rate for all three cases, with slopes at $2.9{\cdot}10^{-3}$ \$/Hz for the Blind-AC, $8.2{\cdot}10^{-4}$ \$/Hz for the LFU-AC, and $1.0{\cdot}10^{-3}$ \$/Hz for the LRU-AC. This confirms that the LFU- and LRU-AC strategies cope better with increased loads (e.g., flash crowds) than the Blind-AC. This is typically due to the improved hit rate at the Fog cache, which absorbs a large part of the

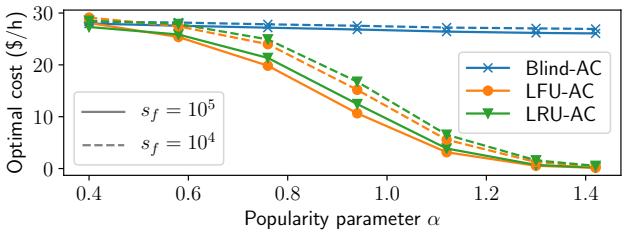


Figure 3.6 – Cost vs Zipf parameter α for the Blind-, LFU- and LRU-AC

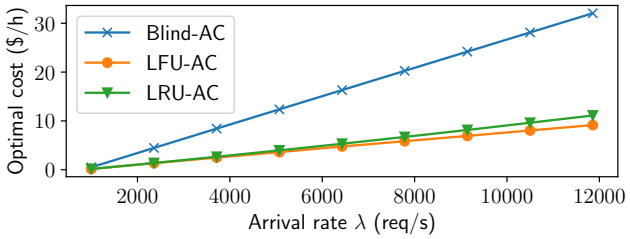


Figure 3.7 – Cost vs arrival rate λ for Blind-, LFU-, and LRU-AC

increased arrival rate. Particularly, if the cost of the Blind-AC diverges with respect to the LFU-AC and the LRU-AC for an increasing arrival rate, their ratio stays however constant. The ratio between the absolute costs for the LFU-AC (LRU-AC) over the Blind-AC is of 3.5 (2.9) [3]. Thus, when the arrival request rate increases, the relative gain of using the LFU-AC (LRU-AC) over the Blind-AC also increases, which shows the LRU- and LRU-AC to be quite robust to high arrival rates.

This preliminary evaluation shows that the LRU-AC combines tractability, flexibility, and effectiveness. It is thus selected as the default admission policy for the remainder of this chapter.

3.5 Ageing Bloom-Filters for an hardware-accelerated LRU-AC

As the admission control strategies were designed to provide predictable completion time under high-load, their implementation should benefit from the same virtues. Thus, implementing the LRU-AC in hardware seems particularly profitable. However, the LRU-AC raises many realization challenges, particularly on hardware-accelerated platforms. The first and obvious one is memory usage. As shown in Section 3.5.4, the total memory footprint of the LRU-AC can overtake the high-speed BRAM (Block RAM [173]) available in the hardware – notwithstanding the BRAM that needs to be allocated for e.g., packet queues. Furthermore, the access time to an LRU element is not constant (as a hash map provides only amortized constant time for read operations), which

3. The relative cost gain of the LFU-AC (LRU-AC) over the Blind-AC ($\Pi_{Blind-AC} - \Pi_{LRU-AC})/\Pi_{Blind-AC}$ is deceptive here as it asymptotically grows to 100%

renders it undesirable for hardware implementation [151]. Thus, in this section, we explore Ageing Bloom Filters (ABF) as an alternative data-structure to implement the LRU-AC. After a brief description of the ABF behavior (Section 3.5.1), a predictive model similar to Che's approximation (Equation (3.2)) for the hit-rate of the ABF is presented (Section 3.5.2). The model is verified by way of simulation (Section 3.5.3) and memory gains compared to standard LRU implementations are evaluated (Section 3.5.4).

3.5.1 Ageing-Bloom filters

A popular and natural structure for storing sets in programmable hardware are Bloom filters [174–176]. Indeed, they provide a compact and efficient way to store set membership with a controllable error probability as the only trade-off. In particular, Bloom filters have been proposed to store network identifier for high-performance packet forwarding [176, 177]. Thus, using a Bloom filter to store the content of the LRU-AC seems like a promising step towards implementing it in hardware. Realizing the LRU-AC, however, requires the ability to evict old content from the cache, which standard Bloom filters do not support. To replicate this behaviour, Yoon introduced Aging Bloom Filters (ABF) [152].

An ABF consists of two parallel Bloom filters: the *active* Bloom filter A_1, used to learn the most recent items, and the *passive* Bloom filter A_2, which holds older items in memory. It has two parameters: n_a, the maximum number of items that each filter holds, and f_p, a target false positive rate. The functioning of the ABF is summarized in Algorithm 1. In steady state, the passive filter A_2 holds exactly n_a different items. The active filter A_1 holds $0 \leq cnt < n_a$ items, some redundantly with A_2. An item is said to be in the ABF if it belongs to either A_1 or A_2. Insertion is done only in the active filter A_1 until it holds n_a unique items. At this point, A_2 is swapped with A_1, and the active filter is reinitialized. This ensures that the ABF contains at least the last n_a different items received, and at most $2n_a$. However, there are redundancies between A_1 and A_2, the exact number of objects in the ABF is a stochastic process.

Algorithm 1 The ABF main functions

```
function INIT                                          ▷ Filter initialization
    A_1 ← ∅, A_2 ← ∅, cnt ← 0
end function
function CHECK(item)                    ▷ Checks if an item is in the filter
    return (item ∈ A_1 ∪ A_2)
end function
function SET(item)                                  ▷ Inserts an item in the filter
    if item ∉ A_1 then
        A_1.INSERT(item)
        cnt ← cnt + 1
        if cnt = n_a then
            A_2.FLUSH()
            SWAP(A_1,A_2)
            cnt ← 0
        end if
    end if
end function
```

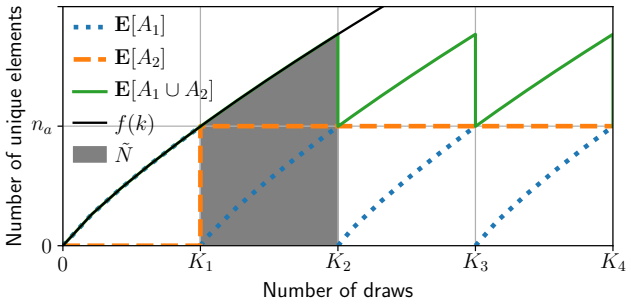


Figure 3.8 – Number of distinct elements in the active filter A_1 (··), in the passive filter A_2 (- -) and in the full filter $A_1 \cup A_2$ (–), as well as the function $f(k)$ (–) used to compute them and the stochastic average $\tilde{N}$ of $|A_1 \cup A_2|$.

3.5.2 Hit-rate approximation for the ABF

To replace the LRU-AC with an ABF, the expected hit-rate of the ABF must be evaluated. Indeed, as shown in Section 3.4.5, the hit-rate distribution is the major factor that explains the performance of the LRU-AC. While for brevity reasons, only the major ideas behind the model derivation are presented here, a complete explanation is provided in Appendix A.

As in Section 3.3, Zipf arrivals under IRM are assumed. Let us consider $|A_1 \cup A_2|(k)$ the stochastic process representing the number of unique elements in an ABF after k arrivals. First, as depicted in Figure 3.8, one can observe that the behaviour of an ABF is cyclic (except for the bootstrap, during which A_2 is empty). At a given time, its behaviour only depends on the arrivals which occurred after the second-to-last flush-and-swap operation as the memory of events that occurred before that point has been flushed. Therefore (and since arrivals are memoryless due to the IRM assumption), it suffices to analyse the behaviour of the filter during the second cycle. Let us denote by K_1 (respectively K_2) the number of steps necessary to complete the first (respectively second) cycle:

$$K_1 = \min\{k \geq 0 : |A_1|(k) = n_a\}$$
$$K_2 = \min\{k \geq K_1 : |A_1|(k) = n_a\}$$

We will thus study the behaviour of the filter in $[K_1, K_2)$.

Approximating K_1 and K_2

For $k \leq K_2$, the probability $h_k(r)$ that content r is in the filter after $k \leq K_2$ draws from the catalogue is simply the probability that it was selected at least once from the k draws:

$$h_k(r) \stackrel{\text{def}}{=} \mathbf{P}[r \in (A_1 \cup A_2)(k) | K_2 \geq k] = 1 - (1 - q(r))^k$$

Thus, the expected number $f(k)$ of items in the filter after k draws is:

$$f(k) \stackrel{\text{def}}{=} \mathbf{E}[|A_1 \cup A_2|(k) \mid K_2 \geq k] = \mathbf{E}[\sum_{r=1}^{R} \mathbf{1}_{r \in (A_1 \cup A_2)(k)} | K_2 \geq k]$$
$$= \sum_{r=1}^{R} h_k(r) = \sum_{r=1}^{R} [1 - (1 - q(r))^k]$$

It is possible to prove, using Chernoff bounds (see Appendix A.2) that K_1 (the number of steps necessary for there to be n_a items in A_1) deviates little from $\widehat{k_1}$, the number of steps necessary for there to be n_a items *in average* in A_1, defined as:

$$\widehat{k_1} \stackrel{\text{def}}{=} f^{-1}(n_a) \tag{3.6}$$

Due to the cyclic behaviour of the filter, K_2 is then naturally approximated as $K_2 \approx 2\widehat{k_1}$.

It is then possible to efficiently find $\widehat{k_1}$ by using an approximation for $f(k)$, rather than inverting a sum with R elements. Assuming a Zipf catalogue with a parameter α (*i.e.*, $q(r) = \frac{r^{-\alpha}}{H_{R,\alpha}}$ with $H_{R,\alpha} = \sum_{i=1}^{R} i^{-\alpha}$), we have:

$$f(k) \approx \begin{cases} Ak - Bk \log k & \text{if } \alpha = 1 \\ Ak - Bk^{1/\alpha} & \text{if } \alpha \in (1/2, 1) \cup (1, +\infty) \end{cases} \tag{3.7}$$

where:

$$A = \begin{cases} \dfrac{R^{1-\alpha}}{(1-\alpha)H_{R,\alpha}} & \text{if } \alpha > \dfrac{1}{2}, \alpha \neq 1 \\ 1 + \dfrac{1 + \log\log R - 2\gamma}{\log R + \gamma} & \text{if } \alpha = 1 \end{cases}$$
$$B = \begin{cases} \dfrac{-\Gamma(-1/\alpha)}{\alpha^2 H_{R,\alpha}^{1/\alpha}} & \text{if } \alpha > \dfrac{1}{2}, \alpha \neq 1 \\ \dfrac{1}{\log R + \gamma} & \text{if } \alpha = 1 \end{cases} \tag{3.8}$$

where $\Gamma(z)$ is the gamma function and $\gamma \approx 0.577$ the Euler-Mascheroni constant. The derivation is provided in Appendix A.3.

Approximating the hit rate

As introduced in the previous section, the hit rate $h_k(r)$ for content r after $k \leq K_2$ draws from the catalogue is:

$$h_k(r) = 1 - (1 - q(r))^k$$

Thus, the (stochastic) hit rate $H(r)$ for content r when observed at an instant K_{obs} drawn uniformly during the cycle $[K_1, K_2)$ is:

$$H(r) \stackrel{\text{def}}{=} \mathbf{E}[\mathbf{1}_{r \in (A_1 \cup A_2)(K_{obs})} | K_1, K_2]$$
$$= \frac{1}{K_2 - K_1} \sum_{K_1 \leq k < K_2} \left[1 - (1 - q(r))^k\right]$$

and the corresponding average hit rate $h(r)$ is:

$$h(r) \stackrel{\text{def}}{=} \mathbf{E}[H(r)] = \mathbf{E}\left[\frac{1}{K_2 - K_1} \sum_{K_1 \leq k < K_2} \left[1 - (1 - q(r))^k\right]\right]$$

Since (as argued in the previous section) K_1 and K_2 can be approximated by $\widehat{k_1}$ and $2\widehat{k_1}$, respectively, we can further provide an approximation $h_a(r)$ of $h(r)$ as:

$$\begin{aligned} h(r) &\approx \frac{1}{\widehat{k_1}} \sum_{\widehat{k_1} \leq k < 2\widehat{k_1}} \Big[\underbrace{1 - (1 - q(r))^k}_{\approx 1 - e^{-kq(r)}}\Big] \\ &\approx \frac{1}{\widehat{k_1}} \int_{\widehat{k_1}}^{2\widehat{k_1}} \left[1 - e^{-kq(r)}\right] \mathrm{d}k \stackrel{\text{def}}{=} h_a(r) \end{aligned}$$

Simple algebra then yields:

$$h_a(r) = 1 - e^{-q(r)t_C(r)} \tag{3.9}$$

where:

$$t_C(r) = \widehat{k_1} + \frac{1}{q(r)} \log \frac{\widehat{k_1} q(r)}{1 - e^{-\widehat{k_1} q(r)}} \in [\widehat{k_1}, \frac{3}{2}\widehat{k_1}]$$

A Che-like approximation

For large values of r, $t_C(r)$ exhibits very little variation: a Taylor expansion shows that $t_C(r) \approx \frac{3}{2}\widehat{k_1} - \frac{q(r)}{24}\widehat{k_1}^2$. For small values of r, $t_C(r)$ varies more, but its contribution to $h(r)$ can be neglected as $h(r) \approx 1$ in those cases, as argued in [178] (figures depicting this phenomenon are available in Appendix A.4). Therefore, it is possible to make the approximation that $t_C(r)$ is a constant t_C, yielding:

$$h_a(r) \approx 1 - e^{-q(r)t_C} \tag{3.10}$$

In that case, by summing over $r \in \{1, \ldots, R\}$, t_C can be computed by finding the root of:

$$\sum_{r=1}^{R} [1 - e^{-q(r)t_C}] = \sum_{r=1}^{R} h_a(r) \stackrel{\text{def}}{=} \tilde{N} \tag{3.11}$$

where $\tilde{N}$ represents the average number of items in the filter, and can be computed with:

$$\tilde{N} \approx \frac{1}{\widehat{k_1}} \int_{\widehat{k_1}}^{2\widehat{k_1}} f(k) \mathrm{d}k \tag{3.12}$$

using the approximation of $f(k)$ provided in the previous section and straightforward integration.

This is similar to the Che approximation (see Equation (3.2)), with the (fixed) size of the LRU cache replaced by the average over time of the "size" (number of distinct items) of the ABF. Functions for computing $\widehat{k_1}$ and n_a depending on k_{LRU} are provided in the Jupyter Notebook associated with this chapter [172]. For instance, k_{LRU}=2.8·10^5 (as in Table 3.4) yields n_a=2.0·10^5.

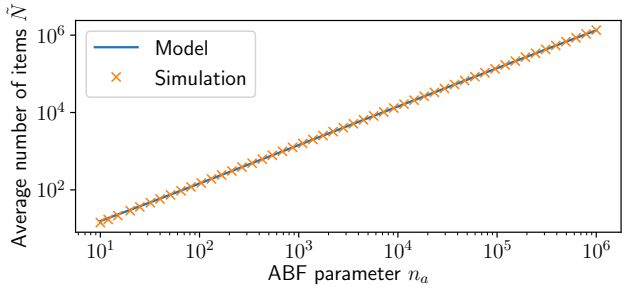


Figure 3.9 – Simulation and model results for $\tilde{N}$ vs n_a (R=10^7 and α=1). Each simulation point consists of the average value on 10^8 arrivals.

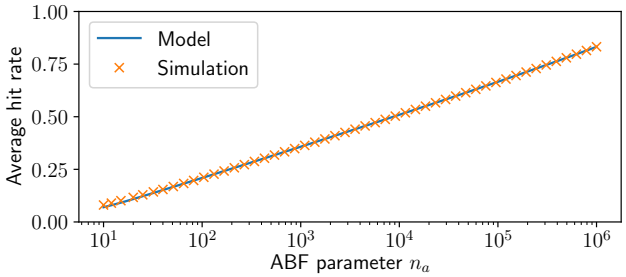


Figure 3.10 – Simulation and model results for the expected hit-rate of the ABF depending on n_a (R=10^7 and α=1). Each simulation point consists of the average value measured over 10^8 arrivals

3.5.3 Model verification for $\alpha = 1$

To verify the model, a software implementation of the ABF was subjected to a Poisson arrival process with Zipf popularity distribution. In particular, in Figure 3.9, measured values of $\tilde{N}$ are compared to the results obtained using the method described at the end of Section 3.5.2. It shows that the analytical model accurately captures the expected number of unique items in the ABF, with an average relative difference 3%. Then, the accuracy of the approximation proposed in Section 3.5.2 is evaluated in Figure 3.10, which compares the average hit-rate $\sum_r q(r)h_a(r)$ computed from the approximation of Equation (3.10) with the measured average hit-rate in simulation runs. It shows that the model fits almost perfectly with the experiments, with a relative difference $< 0.02\%$ for $n_a > 10$.

3.5.4 ABF - memory usage vs LRU

LRU memory

Let us consider a traditional LRU implementation consisting of a hash map pointing to a doubly-linked list. Each LRU entry has a memory footprint at least equal to the size of 3 pointers (one in the hash map and two in the doubly-linked list). The minimal pointer size is $\lceil \log_2 k_{LRU} \rceil$. To compare the ABF

to the LRU and according to Equation (3.11), $k_{LRU} = \tilde{N}$ is used. Thus, the memory used by the LRU is:

$$m_{LRU} = 3\tilde{N}\lceil \log_2 \tilde{N} \rceil \tag{3.13}$$

ABF memory

Let f_p be the wished false-positive rate of the ABF. The corresponding false-positive rate of A_1 and A_2 is $f_a = 1 - \sqrt{1 - f_p}$ [152]. The corresponding number of hash-functions is $n_h = \lceil -\log_2(1 - \sqrt{1 - f_p}) \rceil$, and $m = \frac{2 n_a n_h}{\log 2}$. Putting it all together, the memory consumption of the ABF is:

$$m_{ABF} = \frac{2 n_a \lceil -\log_2(1 - \sqrt{1 - f_p}) \rceil}{\log 2} \tag{3.14}$$

Numerical results

For $k_{LRU} = 2.8 \cdot 10^5$ (as in Table 3.4), the minimal pointer size is 19 bits (since $\log_2(k_{LRU}) \approx 18.1$). The required memory per element adds up to 57 bits. In total, this amounts to $m_{LRU} = 16$ Mbit. By comparison, the corresponding ABF has for parameter $n_a = 2.0 \cdot 10^5$. For a false positive rate of f_p=1%, Equation (3.14) yields $m_{ABF} = 4.6$ Mbit, dividing the LRU-AC memory footprint by 4. Furthermore, let us note that while the computation of m_{ABF} is exact, the computation of m_{LRU} is conservative. Indeed, implementing a hash-table requires a significant memory overhead and creates a memory vs operation-latency trade-off.

3.6 Hardware-implementation of the LRU-AC

3.6.1 Using hICN as the underlying network layer

To guarantee line-rate performances, the hardware-accelerated admission control must be able to access the request identifier easily. While extracting application-level semantics from packets (deep-packet inspection) is possible in hardware using frameworks such as P4 [155], it is costly in terms of cycles. Therefore, the performance of the admission control hardware module would benefit from a network framework that exposes application-semantic at a low-level, but that also provide a deterministic packet format (through fixed-size headers, fixed field locations, such as IP/TCP).

In that regard, hICN [154] is an ideal candidate. hICN is an implementation of ICN [179] that transparently uses the semantic of IPv6/TCP for backward-compatibility. In particular, while hICN is a name-based protocol (i.e., forwarding is performed based on named-content identifiers rather than locators), names are encoded using an IPv6 address for a *name prefix* (64 bits of routable prefix and 64 bits of data identifier) and a 32-bit integer for the *name suffix*. An Interest (resp. Data) packet then carries the name prefix in the IPv6 destination (resp. source) address field, while the name suffix is placed in the TCP segment number field. As such, hICN holds the desirable characteristics for implementing the LRU-AC in hardware: request semantic accessible in *fixed-size* fields in *fixed locations* of the *network and transport headers*.

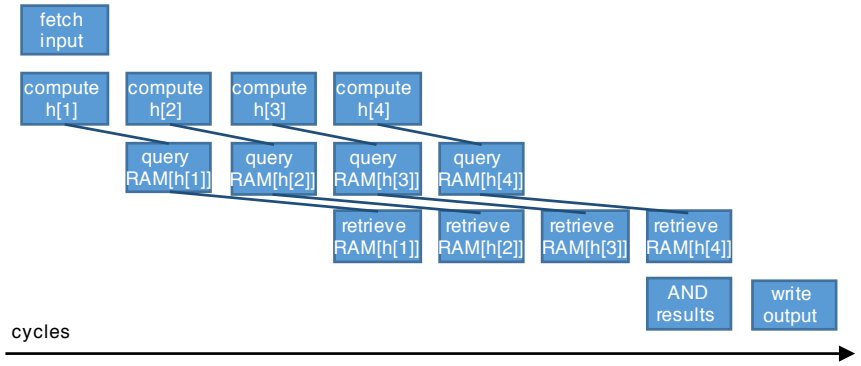


Figure 3.11 – Bloom filter pipeline illustration with $n_h = 4$: latency is 7 cycles, throughput is one operation per 8 cycles.

3.6.2 Hardware-implementation of the LRU-AC

To demonstrate implementability of the LRU-AC in hardware, the NetFPGA-SUME [153], a state-of-the-art academic programmable network cards, was used. To provide a modular and easily-modifiable implementation, the prototype is implemented in P4 [155], allowing packet parsing to be performed in a high-level language. The P4→NetFPGA framework [180] is then used to translate that high-level representation into a NetFPGA-SUME implementation.

The prototype comprises two parts: (i) a Bloom filter *atom*[4] written in Verilog, that implements a single Bloom filter; and (ii) a P4 data-plane, which performs packet parsing, processing, and deparsing, and whose processing part implements an ABF by using two Bloom filters atoms. Using a Bloom filter atom and implementing the ABF logic in P4 rather than directly implementing the ABF as a black-box Verilog module provides greater modularity and expressiveness in the high-level language and simplifies the engineering effort by having to focus on optimizing only a single and simpler low-level module. The Bloom filter atom has been upstreamed to the P4-NetFPGA project[5].

Bloom filter atom

The Bloom filter atom takes a fixed-length key of size s_{key} in argument, as well as an operand specifying the operation to be performed on that key (read or insert), and returns a single bit specifying whether the key was found in the filter – in case of an insert operation, whether the key was found before insertion. Additionally, a third operand allows resetting the filter (*i.e.*, clearing all its bits). This allows exporting a very simple API:

```
extern void bloom_filter(in bit<2> opcode, in bit<KEY_SIZE> index,
    out bit<1> result);
```

The Bloom filter is parametrized by the number n_h of hash functions, and the size s_{hash} of their output – governing the number of addressable objects. Each

4. Atoms are low-level modules that can handle state and perform a simple operation, while interfacing with a higher-level packet processing language [181].

5. `https://github.com/NetFPGA/P4-NetFPGA-public`

Table 3.5 – Resource usage of a Bloom filter atom

s_{hash}	s_{key}	Logic (LUTs)	Registers	BRAM	Multipliers
20	24	185	134	33	2
21	24	246	152	65	2
22	24	357	186	129	2
20	48	281	178	33	4
21	48	317	196	65	4
22	48	431	230	129	4

hash function h^i ($i \in \{1, \ldots, n_h\}$) is implemented using universal hashing [182]. Indeed, universal hashing relies only on multiplication, XOR and shift operations, and can thus be efficiently implemented on the NetFPGA-SUME (using multiplier blocks).

To optimize throughput and latency, the Bloom filter is implemented with a pipelined approach, as depicted in Figure 3.11. The idea is to use each cycle to query one bit (at the address dictated by one of the n_h hash functions), for a total of approximately n_h cycles. Since querying the BRAM has a 2-cycle latency and hash function computation uses a full cycle, the i-th hash function is computed at cycle i and the corresponding bit is queried at cycle $i+1$ for a result retrieved at cycle $i+3$. The final result is then output at cycle n_h+4, after ANDing all the intermediary results. In sum, the Bloom filter has a latency of n_h+3 cycles and a throughput of 1 operation every n_h+4 cycles. In terms of spatial complexity, in addition to the LUTs (Look-Up Tables, the fundamental reconfigurable logic blocks in FPGAs) required to implement the logic, the module uses $2^{s_{hash}-15}$ blocks of BRAM (of size 32 Kbit). Table 3.5 reports the resource usage of a single Bloom filter after synthesis, for different values of s_{key} and s_{hash}.

P4 data plane

The P4 data plane leverages the simplicity of having defined an external Bloom filter atom to provide a simple implementation. It comprises four components: (i) a parser, which extracts Ethernet and IPv6 headers and stores the hICN object key, (ii) an action, which implements the ABF logic to determine the egress interface for the packet, (iii) a match-action table, which maps that interface to an Ethernet address, and (iv) a deparser, which reconstructs the output packet. (i), (iii) and (iv) are straightforward to implement in P4. (ii) is implemented through four external atoms: two Bloom filters, a flag that keeps track of the active Bloom filter, and a counter that keeps track of the number of requests since the last swapping event. Predicated-read-add-write registers available from the P4-NetFPGA framework [181] are used to implement the counter and the flag. The counter is incremented if its value is smaller than $\widehat{k_1}$ and reset when it reaches $\widehat{k_1}$ [6]. The flag is swapped when the counter has just

6. $\widehat{k_1}$ is used as a threshold on the number of steps rather than n_a on the number of elements in the filter because the `reg_ifElseRaw` atom can only be accessed once per packet in the P4-NetFPGA workflow. To use n_a, the counter would have to be read before querying the filters (to decide whether to swap) and updated after the queries have completed (to count the number of active elements). The validity of using $\widehat{k_1}$ comes from Equation (3.6).

been reset. Depending on the value of the flag and on whether it has just been swapped, suitable (read, write or clear) operations are sent to the two Bloom filters, along with the key extracted from the packet. Finally, the output port is chosen, depending on the OR of the result of the two queries.

3.7 Evaluation

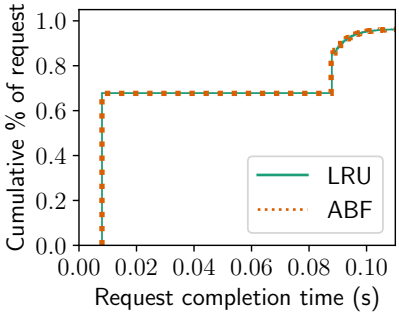
In this section, an evaluation of the FPGA-based LRU-AC (named ABF-AC in this section to distinguish it from the LRU-AC implemented with an actual LRU filter) is presented. First, packet-level simulation is used to compare the ABF-AC to the LRU-, LFU- and Blind-AC. Then, the throughput and processing speed of the FPGA implementation is evaluated.

3.7.1 Packet-level simulation

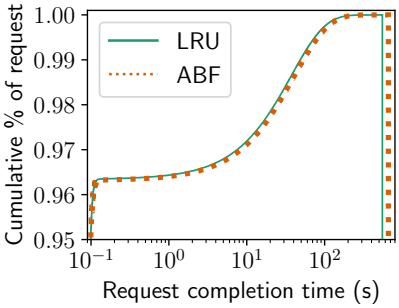
To derive more insights about the packet-level behavior of the schemes introduced in Section 3.4, packet-level simulations of the queueing network introduced in Section 3.3 using actual implementations of each strategy were conducted. The simulator, written in Rust and available in open-source [172], is a general-use queueing simulator designed to allow quick specification and testing of queueing networks. In this section, it is set up with the values presented in Table 3.3 and the admission control modules with the values computed in Table 3.4. The ABF filter is configured with $\widehat{k_1} = 5.2{\cdot}10^5$ (computed through Equation (3.6)) and f_p=1%. The interested reader can use the Jupyter notebook provided with the simulator to find the filter parameters tailored to their own scenario and set up the simulator accordingly.

In a first step, the per-packet response time of the LRU-AC is considered. Figure 3.12 shows the cumulated distribution function (CDF) for both the ABF- and LRU-AC. For the sake of clarity, the head and the tail of the distribution are represented in resp. Figure 3.12a and Figure 3.12b, while individual CDFs for the Fog and Cloud paths are represented in Figure 3.12c and Figure 3.12d. The only visible difference between the LRU- and the ABF-based implementations is the spread of the distribution, which concerns only 0.1% of requests, thus justifying the validity of the ABF-AC. Of further note is the length of the queue, which goes up to 600s. As shown by Figure 3.12, this is due to the effect of the tri-modal nature of the distribution on the problem formulated in Equation (3.1). Indeed, a large percentage of requests are processed with a latency $\ll \Delta$. Thus, the constraint $\mathbf{E}[T] \leq \Delta$ can tolerate a long tail, which might be a problem for real-life applications (even if 99% of the requests are completed under a minute). For operators with stricter latency constraints, the length of the queue can be reduced by slightly modifying the value of k_{LRU} and $\widehat{k_1}$. For instance, artificially reducing the load on the Fog by 5 percentage points by setting k_{LRU}=$1.3{\cdot}10^5$ (resp. $\widehat{k_1}$=$2.3{\cdot}10^5$) allows reducing the distribution spread to about 0.3 s and the threshold excess rate (i.e., % of requests with service time $\geq \Delta$) to 1% for a 16% cost increase for both implementations, as shown in Figure 3.13.

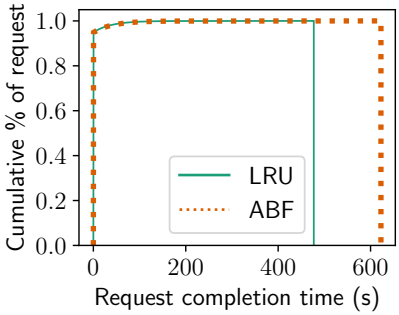
In a second step, Figure 3.14 shows the request path repartition and the threshold excess rate for the Blind-, LFU-, and both implementations of the LRU-AC. It highlights that the threshold excess rate is of the same magnitude between all schemes, with probabilistic LRU-AC just slightly higher. Further-



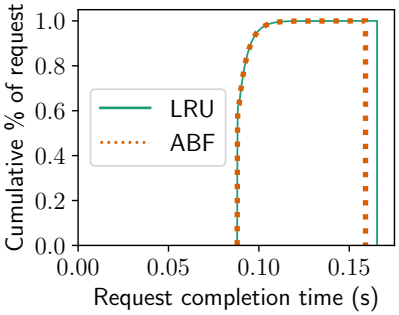
(a) Head (Fog+Cloud)



(b) Tail (Fog+Cloud)



(c) Fog



(d) Cloud

Figure 3.12 – Cumulative distribution functions of the measured request response time for the LRU- and ABF-AC

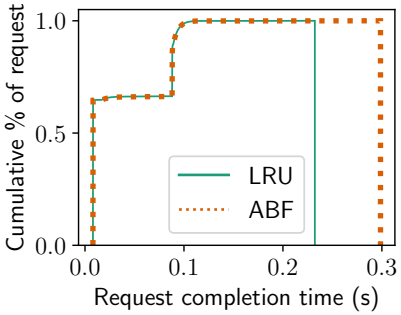


Figure 3.13 – Cumulative distribution function of the measured response time for conservative settings of the LRU- and ABF-AC

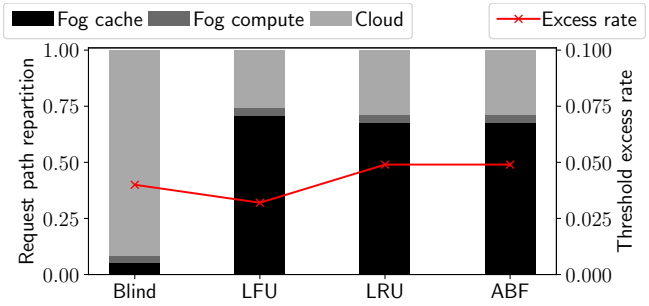


Figure 3.14 – Threshold excess rate (red line, left y-axis) and Request breakdown (grey bars, right y-axis) for the Blind-, LFU-, LRU-, and ABF-AC

Table 3.6 – Dataplane resource usage

	LUTs	BRAM	Power (W)
2 Bloom filters	715	130	0.089
P4	103424	564	8.550
Total	104139	694	8.639
Available resources	433200	1470	—
Resource consumption	24.0 %	47.2 %	—

more, it illustrates again the strong difference between on one side the LFU- and LRU-AC, which handle about 70% of the requests in the Fog, and the Blind-LB, which sends more than 90% of the requests in the Cloud. Popularity-based admission control thus seems an appropriate approach to take maximum advantage of edge resources.

3.7.2 Implementation evaluation

The P4 data plane introduced in Section 3.6.2 has been synthesized for the NetFPGA-SUME platform with the P4→NetFPGA framework. The targeted false positive rate is $f_a = 1\%$, yielding $n_h = 8$. The maximum number of elements in the ABF is taken to be $n_a = 9.2{\cdot}10^4$ (corresponding to an equivalent LRU filter size $k_{LRU} = 1.3{\cdot}10^5$, i.e., the conservative settings introduced in Section 3.7.1), yielding $m_{ABF}/2 = 2.1{\cdot}10^6$ bits in each filter. Thus, one must take $s_{hash} = \lceil \log_2 \frac{m_{ABF}}{2} \rceil = 21$ to be able to address all elements in each filter. Finally, we take a key of $s_{key} = 48$ bits, allowing to address up to $280{\cdot}10^{12}$ objects, way above the catalogue size $R = 10^7$.

Resource usage on the FPGA board is reported in Table 3.6. As can be seen by comparing these results to those in Table 3.5, most of the logic (LUT) is consumed by the NetFPGA framework (MAC for the network interfaces and packet processing). However, an important part of the BRAM resources is consumed by the Bloom filter modules. In total, the BRAM usage is 47%, confirming that the limits of the platform are reached and that a standard LRU filter (which would consume 4 more memory as shown in Section 3.5.4) could not be implemented.

The performance of the P4 data plane is evaluated by injecting in the FPGA

Table 3.7 – Dataplane performance

Latency (μs)	Throughput (Mpps)
2.62	16.7

simulator a batch of 4096 packets (directly after the Ethernet interface, so as to outreach the 10 Gbps limit), and measuring the corresponding latency and throughput. Results are reported in Table 3.7, showing that packets can be forwarded over the 10 Gbps line-rate (14.4 Mpps) while providing low latency. The obtained throughput results are consistent with the throughput of the Bloom filter atom (one operation every $n_h + 4 = 12$ cycles at 200 MHz).

3.8 Related Work

The importance of locating computing resources topologically close to users has been put forward under diverse forms in the community: Fog computing [17], Mobile-Edge-Computing [183], hybrid Cloud [149]. In particular, Niu et al. [149] explore a similar problem to ours: the use of a local Cloud infrastructure to handle sudden bursts of traffic. They also use a Markov-chain based model for computing the expected completion time and devise a scheduling algorithm between hybrid and public Cloud using an optimization problem under budget constraints. However, they do not exploit any knowledge of the request popularity, thus falling under the hard limit that we exposed for the Blind-AC. Malawski et al., study costs optimization between a hybrid Cloud and multiple public Clouds with different pricing models under a deadline constraint [184]. However, they focus on task optimization, looking at a model closer to scheduling for scientific computing rather than live optimization of user requests. Du et al. [150] formulate a joint resource allocation and offloading between user devices, Fog networks and Cloud networks, so as to minimize energy consumption and request processing delay.

Using popularity to load-balance content in ICN networks has already been explored. In [157], the authors propose to count incoming packets and use exponential smoothing. As argued in Section 3.4.4, this approach is not flexible to popularity changes and requires knowledge of the application. Furthermore, the authors aim at load-balancing packets over homogeneous paths, whereas the Fog/Cloud offload problem is essentially heterogeneous. Similarly, Carofiglio et al. [158] propose to use a k-LRU filter to learn popularity for load-balancing ICN interests over multiple paths. They then measure per-name latency to optimize the distance to the next object. However, the authors do not specify the settings of the k-LRU filter, and only consider the effect of their load-balancing on the data creation process. Finally, in [71], the authors use the ICN-Fog node as a classifier between static and dynamic data, thus preventing upstream caches to store dynamic data. They do not, however, consider the data processing that happens in many Fog applications.

3.9 Summary

In this chapter, we introduced methods for guaranteeing response time in Fog deployments based on popularity-aware admission control. Two specific admission control schemes were defined: the oracle-based LFU-AC and the probabilistic LRU-AC. Their effectiveness was demonstrated using an analytical model for various application parameters. An implementation of the LRU-AC on state-of-the-art FPGA hardware using an ABF was then proposed and its soundness was demonstrated through analytical modeling. This implementation is shown to provide admission control at 10 GbE line-rate throughput with a 3 µs latency. It increases the acceptance rate by almost 10 w.r.t. content-blind approaches while maintaining the latency excess rate stable.

Chapter 4

Network and application management in Information-Centric Networks

As illustrated in the previous chapter, one of the big advantages of ICN is the tight integration between applications and the network. In this chapter, we argue that to realize the full potential of such a cross-layer collaboration, most of the network tooling must be rethought accordingly. This is particularly true for the management of both applications and networks, and all-the-more for platforms such as Fog computing, which are highly virtualized [17]. Management frameworks should reflect the characteristics of ICN, for instance by intertwining network and application management. We thus attack the network management problem from a general and high-level perspective. In particular, we explore the relation between ICN and Intent-Based Networking (IBN), a network management design where administrators only specify an abstract view of their desired network model instead of detailing the necessary steps to attain it. We explore both sides of the problem: how can IBN be applied to ICN but also how ICN can be used to enhance IBN.

4.1 Intent-Based Networking and ICN

Recent years have seen the emergence of network paradigms such as network function virtualization (NFV) or Cloud and Fog computing, an increasing complexity of network security requirements, and a shift in network communication patterns. Human operators are thus struggling to keep up with the complexity and diversity of network devices, services and traffic. In particular, network management has essentially become a multiparty process, where deploying a service involves many actors. This has led to the development of Software-Defined Networking (SDN) [185] and the apparition of feature-rich, centralized network and service orchestrators such as OpenStack [186] or Kubernetes [187], built to present simplified and unified human-facing interfaces. Of particular note is the switch from *procedural APIs* to *intent-based* network configuration [188, 189], where the user specifies an intent in a human-readable and abstract way, e.g.,

"*Deploy a video-conferencing service for my company*", instead of the full concrete procedure necessary to achieve it. Intent-based networking is touted as an enabler for complete network automation, where a self-regulating network is able to digest and apply high-level policies without human interaction [190]. In fact, with its stated aim of abstracting the actual configuration in favour of abstract policies [188, 189], intent-based networking raises many research challenges, e.g., how to use artificial intelligence on telemetry data to verify and optimize policy application [191].

Progress on IBN has not yet been reflected in ICN. The ICN community has admittedly stated the importance of a pragmatic experimental and application-driven research approach since its inception [40, 192]). Multiple tools and testbeds have even been developed for simulation and emulation (CCNx, NDN software and testbed, CCN-lite, MiniCCNx [193], MiniNDN). Most of them have however been designed to assist research, specifically on the design and evaluation of specific aspects of the ICN architecture (e.g., caching, forwarding, or routing). They operate in dedicated fully-ICN network environments, trading-off abstraction of network characteristics for scale and offering limited flexibility to modify core ICN features, network topology and settings, or application APIs. To bridge the gap between a promising network architecture and a feasible deployment-ready solution, problems such as network configuration, management, and orchestration must be resolved. On the other hand, ICN offers the opportunity to rethink IBN frameworks, looking not only at how IBN should be applied to ICN but also at how ICN can enhance and empower intent-based network management.

First, we present *vICN (virtualized ICN)*, a flexible intent-based framework for ICN network configuration, management, and control that is able to satisfy a number of important deployment and experimentation use cases: (i) conducting large-scale and fine-controlled experiments over generic testbeds; (ii) instantiating reliable ICN network with real applications in proofs of concept; (iii) deploying large networks within networks for trial and test development. Clearly, requirements are different: research experimentation needs fine-grained control and monitoring of the network as well as reproducibility of the experiments. Prototypes for demonstration require a high level of programmability and flexibility to combine emulated and real network components or traffic sources. More than in previous cases, reliability and resource isolation is a critical property for deployments in ISP networks.

The operations required for the deployment of an ICN network include installing/configuring/monitoring a new network stack in forwarding nodes or the socket API used by applications at the end-points. If loading a cloud microservice from an application store into general purpose hardware is easy to realize, a network stack has different requirements: ultra-reliability, high-speed and predictability, to cite a few. vICN shares the same high-level goals as SDN/NFV architectures, but with additional ICN-specific capabilities not typically required by IP services. Overall, we identify three main challenges that vICN addresses and that differentiate it w.r.t. state of the art:

Programmability: i.e., the need to expose a simple and unified intent-based API, intuitive enough to facilitate bootstrap, expressive enough to accommodate both resource configuration and monitoring, and flexible enough to allow the user to decide about the level of control granularity. Exist-

ing software like OpenStack, which is built as a collection of independent components, each one following different design patterns, does not offer such a satisfactory level of programmability.

Scalability: vICN aims at combining high-speed packet processing, network slicing and virtualization, and highly parallel and latency minimal task scheduling. Current systems are based on a layered architecture that prevents fine-grained optimization, thus limiting scalability on the long term.

Reliability: a fundamental property of vICN lies in its ability to maintain the state of deployment, recover from failures and perform automatic troubleshooting. This requires the overall software to be able to accommodate programmable function monitoring and debugging. In existing designs, each component has independent implementations to achieve that.

In a last step, and looking ahead to future research directions, we turn the tables and investigate how an ICN-like approach can benefit IBN. In particular, we argue that many limitations of current orchestrators come from the necessity to perform an early and centralized binding from the user's request to a set of device configurations, requiring full knowledge and preventing any further actors to participate in the resolution of the intent. We thus propose to push the intent deeper into the network fabric rather than limiting it to the edge, which means both transporting and processing it in-between network elements or orchestrators. Inspired by ICN, we introduce an Intent-Centric network management protocol, used to *route intent* in a distributed fashion rather than *pushing configuration* from a centralized node. We propose a router architecture as well as tentative ideas for forwarding algorithms. We show how carrying intent over the management plane could simplify orchestration systems, providing scalability, reliability, and flexibility by offloading many of their central functions.

The work presented in this chapter has been the object of three publications. [101,104] cover the design and implementation of vICN. [105] focuses on making the case for an Intent-Centric network management protocol. Furthermore, the vICN orchestrator has been released as open-source as part of the Linux Foundation Fast-Datapath (FD.io) project [194].

The remainder of the chapter is organized as follows: Section 4.2 summarizes the state of the art, before introducing the vICN architecture in Section 4.3 and its implementation in Section 4.4; Section 4.5 provides concrete examples of vICN in action; Section 4.6 looks at future research directions, in particular how ICN-like approaches can be applied to create truly autonomous networks; Section 4.7 concludes this chapter.

4.2 Related work

Salsano et al. [195] propose to introduce network virtualization for ICN networks through the use of OpenFlow. In [196], the authors address a similar issue and propose an architecture to perform network slicing. However, [195] does not consider any network slicing technology while [196] misses the aspects of network management and control. Mininet [197] makes a step in that direction and sits closer to our objectives as it enables the creation of virtual networks based

on containers and virtual switches. Application performance can be tested in emulated network conditions by setting parameters such as link delay and capacity, node CPU share, etc. However, it does not propose any slicing mechanism and lacks support for wireless or any control on applications or workload.

Interesting tools have also emerged from the testbed community. Emulab [198] is a network experimentation framework joining emulation facilities with physical testbeds, but it lacks support for wireless topologies and offers no control over the network resources. NEPI [199] is maybe one of the most polyvalent tools. It hides all the complexity under a uniform programming interface. NEPI, however, lacks some control granularity and specializes in the management of resources provided by testbeds, assuming tasks such as slicing are already performed.

The Cloud computing community has made important efforts to facilitate the use of data center resources. Cloud Operating Systems have been proposed, such as OpenStack [186], designed to manage and monitor large-scale deployments, providing access to network, computing, and storage resources through a set of homogeneous APIs and sub-projects. Available tools are generally oriented towards applications being deployed in a global pool of resources. Container-specific tools such as Kubernetes [187] present some interesting aspects in that they expose a unique consistent API for simplicity, with however limited control over granularity for our purpose. Automation is ensured by third-party tools layered on top of these standard APIs, like Chef [200], which are intrinsically limited by their procedural language design, where the user must make every step of the deployment explicit, manually adapt to the current state, and handle errors. Other tools (e.g., Puppet [201]) use a descriptive language, where the user only needs to describe his/her needs and leave the rest to the tool. Despite the integration effort in Cloud computing, the silos around functionalities and the proliferation of APIs appear limiting for our purpose. The system does not enable simple setup and control for users, nor to build applications on top.

The now joint SDN and NFV communities are maybe the closest to our needs, at least from an architectural point of view. In [202] for instance, the authors describe a set of design principles for the Management and Orchestration (MANO) of virtualized network functions (VNF). Their approach is based on a modular architecture, clearly identifying the fundamental function such as user and VNF description, orchestration, etc. They also point the need to ensure reliable management by considering the lifecycle of the resource they manage. OpenDayLight is a promising candidate framework for building NFV capabilities, as it relies on a model-driven abstraction layer that fits with our requirements. However, this aspect is mainly used from a software engineering point of view rather than to offer programmability of the resource (called "micro-service"). Moreover, the orchestration is layered on top of other modules that behave as silos.

4.3 The vICN framework

The high-level architecture of vICN, presented in Figure 4.1 reminds of NFV proposals such as MANO. Our contribution is in the design of a resource model able to carry intent and on its use as the underlying end-to-end unifying language.

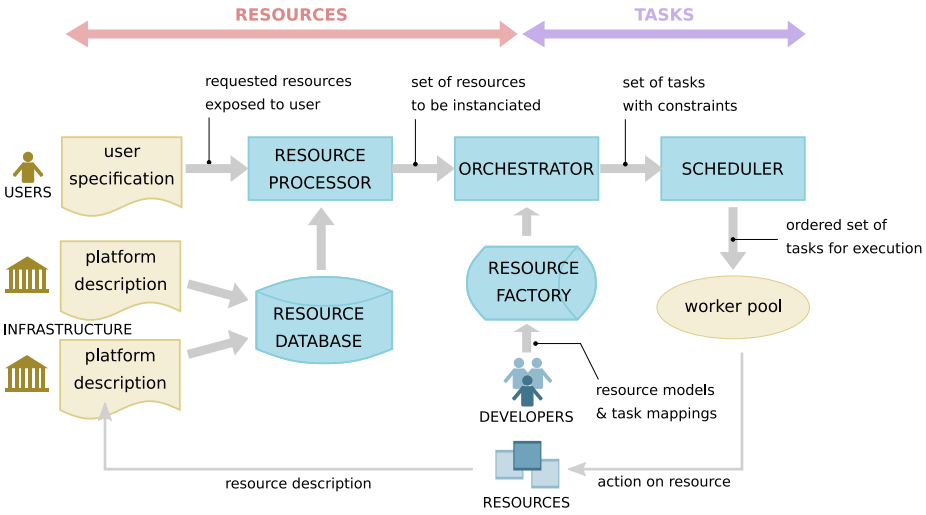


Figure 4.1 – vICN functional architecture

4.3.1 Functional architecture

The Resource is the basic unit of information in vICN. It consists of an abstract model that holds at the same time the current network state and the desired model. Resources are stored in the **Resource factory** and are diverse in type (e.g., a specific Virtual Machine, an application or an IP route). They can be combined or extended to form other Resources. The vICN architecture differentiates the role of users, developers, and infrastructure providers. Developers integrate tools by creating new Resources or extending the old ones. Both users and infrastructure providers use this base set of Resources to describe what they respectively require (users) or make available (infrastructure providers).

Resources are specified according to different degrees of detail: e.g., the infrastructure is described precisely to form a **Resource database** that serves as a base for deployment. On the other hand, user specifications might be general or abstract and only mention Resources of interest for the user. The role of the **Resource processor** is to turn an abstract and incomplete description into a set of Resources mapped on the infrastructure leading to a consistent deployment. Once Resources are selected, the **orchestrator** translates them into a set of actions to be performed, based on the current state of the deployment and on constraints due to task synchronization or sequentiality. The resulting actions are processed by a **scheduler**. It outputs an execution plan and dispatches parallel tasks to a worker pool with the objective of minimizing the deployment time.

To summarize, vICN is based on two main abstractions: (i) *Resources*, which are external units of information exposed to users, developers and infrastructure providers; (ii) *Tasks*, which are internal units of information defined by developers to translate Resource requests into changes in the network deployment state.

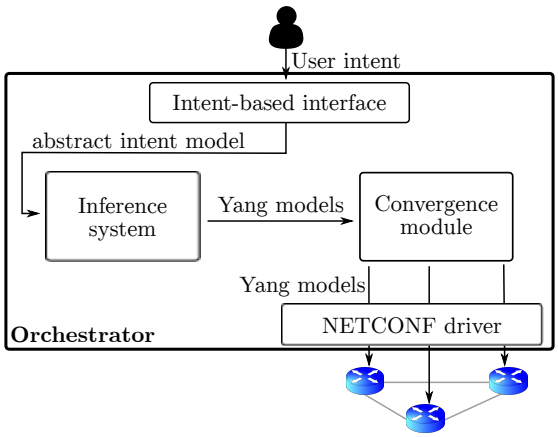


Figure 4.2 – Functioning of current intent-based frameworks

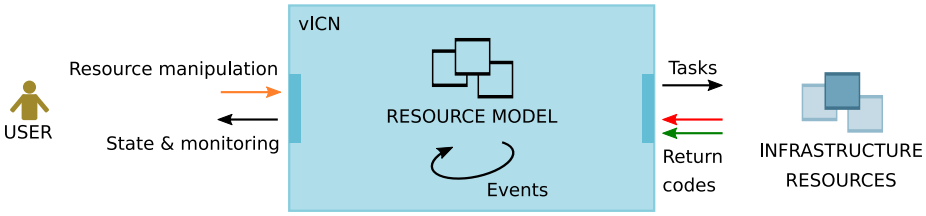


Figure 4.3 – Flow of information in vICN

4.3.2 Resource model

Most of the current intent-based frameworks are built on an intent-model, such as NEMO [203] or PGA [204]. As depicted in Figure 4.2, the role of the Intent module is to translate the intent-model into the configuration model used by the underlying orchestrator, e.g., YANG [205]. It is the configuration model which is used, e.g., starting from the orchestrator module in Figure 4.1. In vICN, a unique Resource model is used globally from the user interface to the scheduling module.

This model, specifically constructed to hold intent rather than configuration, is built using an *object-relational model* [206]. It defines a base *object* as a set of typed attributes and methods, where types refer to standard integers, strings, etc., or to newly defined object themselves. The query language built on top of it (in the spirit of SQL or SPARQL) is used to create, destroy and manipulate those objects, either for Resource setup or to retrieve monitoring information. This model thus benefits from the power and expressiveness of the relational algebra [207] and of some key concepts of Object-Oriented Programming, namely composition and inheritance. It serves as an integrated interface based on both human- and machine-readable semantics.

Resources

Resources in vICN are logical representations of physical and/or remote elements whose state has to be kept synchronized. The state of a Resource can

be affected by user queries, by events involving Resources, or through monitoring queries issued to the remote Resource. For instance, a routing component might recompute routes when notified about a change in the set of nodes, interfaces or links. The corresponding information flow is shown in Figure 4.3.

Such an approach is similar to the IETF standard YANG [205], a human- and machine-readable data model to convey configuration inspired by Object-Oriented programming. However (and importantly), our model differs from YANG, adding three key features to move from holding configuration to modelling intent:

Abstraction: Like [208], we extend YANG with abstract objects by proposing standard models for abstract services (e.g., DNS server, node, or relational database) and resource inheritance. This brings two advantages: it unifies both configuration and intent models, simplifying the translation process, and enable late resource specialization (e.g., from node to Linux Container or Xen VM) in subsequent orchestrators or even at end-devices.

Foreign models: As such, YANG models only describe per-device configuration, which prevents, for instance, device-to-device cooperation to offload part of the scheduling from the orchestrator. The vICN model, in contrary, has the notion of abstract *foreign* objects. For instance, a DNS server can be made aware of the existence of a new node for which it must record a domain name and retrieve the assigned IP address directly from the device on which the node is deployed instead of relying on the orchestrator.

Scheduling information: Foreign requirements are not enough on their own to enable efficient device-to-device cooperation. Indeed, it still presents a flat temporal model while certain services are ordered (some services require others to be completed before them, e.g., a domain-name registration comes after IP address assignment). We thus enrich YANG `groupings` to distinguish *parallel* and *sequential* groupings.

Resource state

The state of a Resource is tracked, for reliability and consistency, by a Finite State Machine (FSM) presented in the left part of Figure 4.4. The FSM models the possible states of a Resource (rectangles, raising events) and the pending operations, or tasks, being executed (round shapes). The transitions are dictated by user actions or internal events and follow the typical lifecycle of an object: *INITIALIZE* is called when the shadow Resource and its object are being created for internal setup; *CREATE* and *DELETE* are the respective constructor and destructor: they can create or destroy the remote Resource and eventually set some attributes; *GET* retrieves the current state of a Resource, as well as the state of some of its attributes; *UPDATE* proceeds to attribute update and, in fact, runs parallel instances of an attribute-FSM, as shown on the right-hand side of the figure.

Resource mapper

For each transition between states, a developer can associate commands to be executed thanks to Resource mappers. These commands are handled by vICN through the task abstraction, which also inherits from the base object. They are specialized to cope with multiple southbound interfaces such as NET-

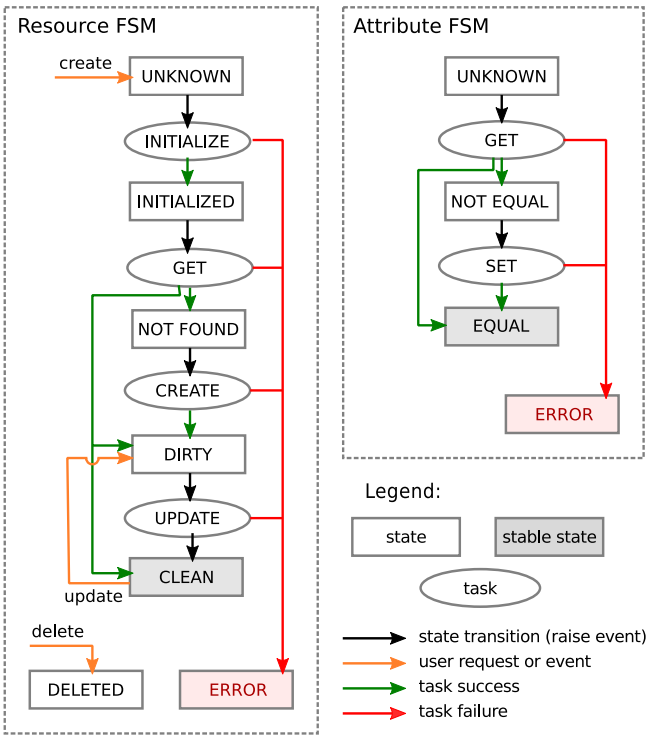


Figure 4.4 – vICN Finite State Machine

CONF/YANG, SSH/Bash or LXD REST calls (similarly to Object-Relational Mappers such as SQLAlchemy [209]).

New tasks can be created through inheritance or composition, using algebraic operators to inform about their parallel or sequential execution. Resource objects are equipped with similar operators so that inheritance and composition produce a similar composition of tasks. Both resources and tasks define an algebra that the scheduler is able to use to perform calculations and optimize the execution plan. A subset of Resources defined in vICN is represented in Figure 4.5, showing, in particular, the four base abstractions of Node, Interface, Channel and Application from which most Resources inherit, similarly to the model defined in [210].

4.3.3 Resource processor

The resource processor plays the central role of adapting the user requests to the platform policies and the available Resources. For instance, an abstract Resource `Node` can be implemented either as an `LXCContainer` or a `VM`. This choice (*specialization* step) can be either explicitly dictated by user preferences or *inferred* by the tool itself depending on the context. As another example, when deploying an ICN forwarder on a node that runs `ndnping`, vICN might prefer an NDN forwarder instance, and do the same for all nodes of the same experiment.

The Resource processor is also in charge of *mapping* the Resources to de-

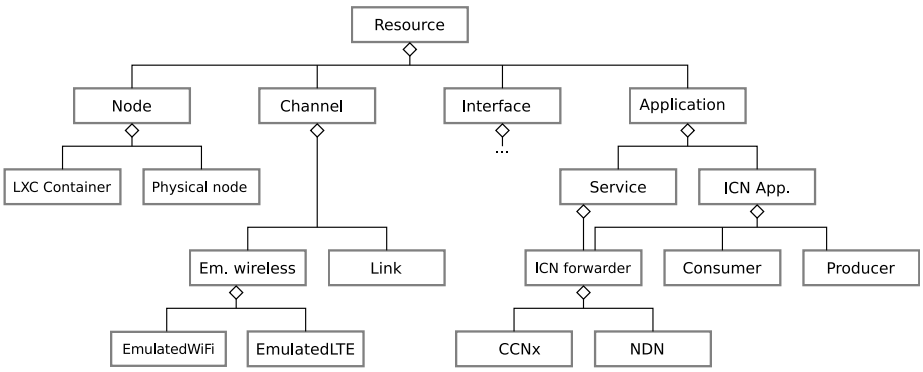


Figure 4.5 – vICN partial Resource Hierarchy

ploy onto available physical servers, by verifying all the constraints/policies required by the user, the developer, or by the infrastructure provider. Such an assignment can be assimilated to an (NP-Hard) Constraint-Satisfaction Problem (CSP) [211] as the system has to accommodate several Resources in a finite capacity system in terms of networking, compute and memory. Both user-specified attributes and optional infrastructure provider policies are taken into account in the CSP as additional constraints. The output is a mapping from specification to implementation, which is also used to expose back monitoring to the user in a consistent way.

4.3.4 Orchestrator and Scheduler

The role of the orchestrator is to maintain one FSM per Resource, and ensure they reach the state requested by the user. Its outcome is a task dependency graph, which is shared with the scheduler. Task dependencies are derived from Resource dependencies, from the structure of the FSM, as well as from inheritance and composition constraints related to both the Resources and the mappers.

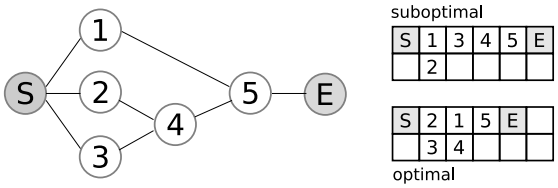


Figure 4.6 – Toy scenario - vICN scheduler

The scheduler ensures the scalability of the deployment by scheduling the parallel execution of tasks over a pool of worker threads. Given the dependency graph presented before, this corresponds to a classical DAG scheduling problem, which has been studied in the community [212]. Figure 4.6 presents a toy-scenario underlining the need for optimizing the scheduling algorithm. In that small example, a greedy selection of the task with the higher distance to the destination is a sufficient heuristic to get an optimal solution and save one

execution round. We remark that user interactions can cause the graph of tasks to evolve in time and require a recomputation. Heuristics [213] might then be preferred to optimal solutions because of their faster execution time, while providing satisfactory performance.

Because of its centralized architecture, the performance of vICN is also impacted by the network transmission time [1]. We alleviate this issue by enabling task batching when two consecutive tasks target the same node interface. The algebraic structure of tasks also makes it possible to reorganize the graph structure to better optimize execution or to increase the ability to batch tasks.

4.4 Implementation

The flexibility of vICN lies in its modular architecture organized around its Resource model. Various Resources can be developed to cover a wide range of underlying infrastructure and bring missing functionalities such as slicing or topology management. We here describe the current release and its set of base Resources covering the whole ICN stack. They build on and reuse available technologies, selected with scalability and reliability in mind.

4.4.1 vICN codebase

A first version of vICN has been open-sourced within the Community ICN (CICN) project [41], as part of the Linux Foundation's Fast Data I/O effort. The code, written in Python, is released under the Apache v2.0 license. This release implements all the building blocks described in Figure 4.1 and is mature enough to launch complex ICN deployments. Alongside, we distribute a prepackaged container image containing the full CICN suite (including forwarders, the ICN stack, and useful applications), so that a full ICN network can be bootstrapped in tens of seconds or minutes. This suite includes a high-speed forwarder based on the VPP framework [214], which already handles almost 1Mpps per thread in its first release.

4.4.2 Slicing

In addition to bare-metal deployments, vICN is able to slice nodes and network links offered by the infrastructure through a set of technologies described here. This is crucial for proper experiment isolation, and to realize separate control and management planes.

Virtual nodes can be implemented either as containers or as virtual machines. We chose containers as the core technology (via the use of Linux Containers [215]) because they are more lightweight and efficient (thanks to zero-copy mechanisms, the ZFS filesystem and simplified access to the physical resources). Increased security concerns and limitations such as sharing the same kernel are not limiting since most ICN functions are implemented in userland.

The network is shared at layer 2 via OpenVSwitch [216], which provides advanced functionalities, such as VLAN and OpenFlow rules, required by our wireless emulators and to bridge real external devices to the virtual environment. vICN fully isolates the deployment's network from the outside world by creating

1. network round-trip-time and, for instance with TLS, session establishment time

a single and isolated bridge per deployment, using iptables as a NAT to provide external connectivity. On top of that, we reduce the load of the bridge and isolate control traffic from the data plane. Indeed, we directly link connected containers, through pairs of Virtual Ethernet interfaces (veth), thus bypassing the bridge. Connected containers that are spawned across different servers in a cluster are transparently connected through a GRE tunnel.

Finally, vICN has to arbitrate for shared resources on the physical host, be it names for interfaces (with constraints such as the 16-character limit on Linux) or containers, VLAN IDs, and even MAC or IP addresses depending on the required level of network isolation. It is important to do such "naming" properly not only for correctness, but also to simplify debugging and troubleshooting. vICN further enforces consistent names that uniquely identify a Resource, which allows for faster detection and recovery when the tool restarts or has to redeploy the same experiment.

4.4.3 IP and ICN topologies

Using the mechanisms described previously, it is possible to build a layer-2 graph on top of which vICN can set up IP and ICN connectivity. For IP networking, a centralized *IP Allocation* Resource is in charge of allocating IP prefixes and addresses to the different network segments of the graph. Global IP connectivity is then ensured by computing the routes to be installed on the nodes. vICN provides a generic *routing module* implementing various algorithms (such as Dijkstra or Maximum-Flow) taking as an input a graph (layer-2) and a set of prefix origins (allocated IP addresses). It outputs a set of routes, encoded as vICN Resources. Route setup is then driven by a *Routing Table* Resource, from which the Linux and VPP routing tables inherit.

The process is similar for ICN, except that we first build (IP or Ethernet) faces based on a configurable heuristic (e.g., L2 adjacency). We can then reuse the same routing module by feeding it with the face graph, and the set of prefix origins found in attributes of content producer Resources. The corresponding Routing Table is, in this case, implemented by the ICN forwarder. We remark that the use of multipath routing schemes (e.g., Maximum Flow) makes more sense in this context. The process results in a deployment accommodating IP and ICN coexistence, enabling performance comparison of both architectures at the same time.

4.4.4 Link emulation

A feature that is missing from most tools is the ability to measure the performance of applications running on top of virtual networks with specific bandwidth or propagation delay. vICN offers Resource attributes for the Linux Traffic Control layer (tc) in order to shape link bandwidth and emulate constrained networks.

A complementary aspect is the ability to use emulated radio Resources as an alternative to real hardware in a transparent fashion. Two types of radio channel are currently supported, WiFi and LTE, both based on real-time simulation features of the NS-3 simulator. The vICN radio channel Resource is implemented as a drop-in replacement of a regular radio link Resource. It connects stations

and access point (or UEs and Base Station) through a configurable radio channel and hides the internal wiring from the user. The emulation then takes care of all relevant wireless features such as *beaconing, radio frequency interference, channel contention, rate adaptation and mobility.* Real-time emulation scales by using multiple instances orchestrated by an overarching mobility management Resource in vICN, communicating in real time with the different emulators. This process can collect relevant information from the simulation, and expose it to the internal model and thus monitoring.

4.4.5 Monitoring capabilities

Monitoring is natively implemented as part of vICN as a transversal functionality, building on the object model introduced in Section 4.3. The language offered by vICN allows querying any object attribute, including annotations made by the Resource processor and orchestrator about the host or the deployment state of the Resource. This is the same interface that is used by the orchestrator to query the current state of a remote Resource, to communicate with the emulators, or for the user to interact with vICN in order to change an attribute or create a new Resource at runtime. Its syntax closely matches the SQL syntax. More precisely a query object contains the following elements: the object name, a query type (create, get, etc.), a set of filters and attributes, eventually completed by attribute values to be set.

For periodic measurements such as link utilization, vICN provides a daemon that can be installed on the nodes and that exposes information via a similar interface. Communication between the components is ensured using the IP underlay setup by vICN.

4.5 Examples

We now illustrate some characteristics of vICN using a particular use case: mobile video delivery. This section is not meant to be exhaustive, but to illustrate how the design of vICN helped us solve practical challenges, and to emphasize general properties of the design that are relevant to other use-cases.

4.5.1 Use case description

The recent years have seen drastic changes in the video consumption patterns that put much pressure on delivery networks: the shifts in video quality (up to 4K), from broadcast to on-demand and from fixed to wireless and mobile networks. Our objective was to show that ICN addresses these challenges, using mechanisms like caching or multihoming over heterogeneous networks. Figure 4.7 represents an example of such a video delivery network. It consists of four parts: a heterogeneous WiFi/LTE access network with multihomed video clients; a backhaul network aggregating the resulting traffic with workload from emulated clients; a core network composed of two nodes; and producers serving 4K video. All nodes have a fully-featured ICN-stack. The core nodes use a VPP-based high-speed forwarder, the others a socket-based one. Overall, the deployment consists of 22 LXC containers, 3 real devices connected to the virtual network, 22 emulated links (including WiFi and LTE channels), and one

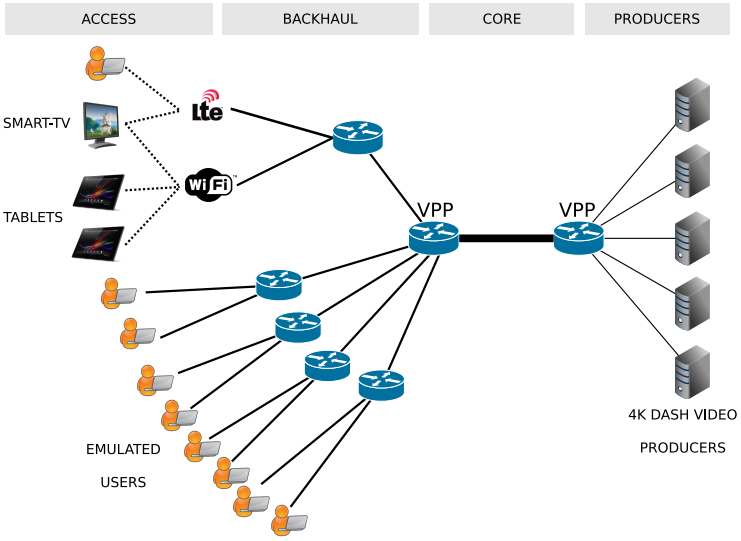


Figure 4.7 – Mobile World Congress topology

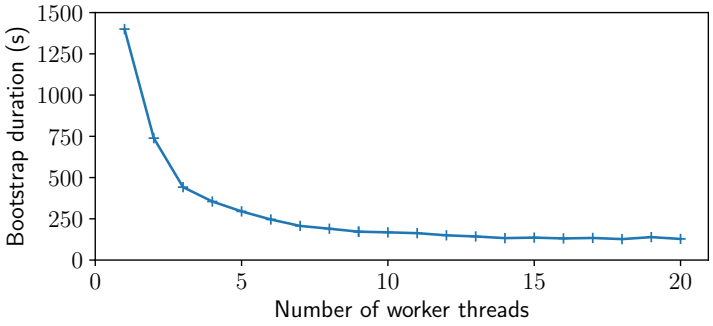


Figure 4.8 – vICN bootstrap time vs number of worker threads

physical link between DPDK-enabled network cards (the core). We use a pre-packaged container image containing all the necessary software to reduce the bootstrap time.

4.5.2 Scalability

The simplification offered by vICN is illustrated by the following numbers: during the deployment, vICN created about 800 Resources compared to the 104 declared in the topology file, a reduction in complexity of 85-90%; more than 1500 bash commands were executed, either directly on physical machines, or on LXC containers. This even underestimates the number of Bash commands an operator would type to deploy an equivalent topology, as some are batched for efficiency reasons (e.g., we insert all IP routes for a given node in a single command).

Figure 4.8 then shows the time taken by vICN to deploy the topology as a function of the number of dedicated threads. We deployed this topology

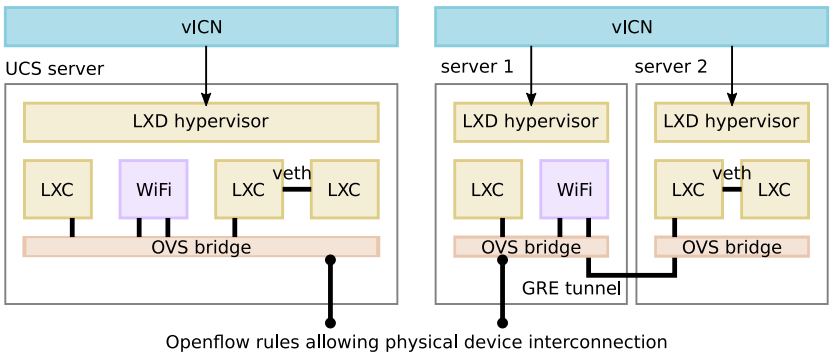


Figure 4.9 – Alternative vICN topology deployments on single server and a cluster.

on a Cisco UCS-C with 72 cores clocked at 2.1 GHz. We first note that multi-threading provides a sevenfold reduction in bootstrap time and that the topology can be deployed in about two minutes. The observed gains are due to the I/O-intensive nature of tasks, which spend most of their lifetime waiting for return values. This reduction is specific to our implementation and our simplistic scheduling heuristic. The shape of the curve remains nonetheless interesting, with a performance bound appearing. This is due to the underlying task graph, whose breadth intrinsically limits the number of tasks that can be run in parallel.

4.5.3 Programmability

One advantage of our Resource model (see Section 4.3.2) is the use of inheritance. It allows the user to choose his level of granularity depending on their needs and expertise. In particular, the user can remain oblivious to the underlying technology used to deploy Resources. We used that feature to scale the demonstration on a cluster of servers connected through a switch instead of a single powerful server. In that configuration, linking containers on different hosts requires to connect them to virtual bridges on their respective hosts, and to link these bridges through an L2-tunnel. The two deployments, shown in Figure 4.9, require different Resources and tasks. However, they can be realized with the same vICN specifications, thanks to the Link abstract Resource. Here, vICN completely abstracts the implementation complexity and enables painless switching from one deployment to the other.

The deployment of containers running VPP is another example of vICN's ability to shield a user from implementation and configuration details thanks to its Resource model. Indeed, VPP uses Direct Memory Access (DMA) to read and write in contiguous memory areas named hugepages. Both the host and the containers have to be configured to allocate and share enough of these hugepages. On top of starting and setting up the application on the container, VPP thus requires to execute commands on the physical node and to change the container's configuration before its creation. In vICN, simply linking VPP to a container is enough to perform the bootstrap. The tool is then able to change the other Resources (e.g., use a VPP-enabled container instead of the standard one) and to run all the necessary commands.

The flexibility of the framework also allowed us to switch Resources in many situations. During our tests, we replaced real tablets by emulated nodes to generate test workloads. During the demonstrations, we could also seamlessly use a real LTE mobile core instead of an emulated one. It only required to change one Resource in the specification and did not affect the rest of the scenario.

4.5.4 Monitoring and Reliability

We conclude by highlighting how the Resource model enables monitoring and debugging. As described in Section 4.4.5, vICN exposes a query language based on its underlying model for monitoring. This language can be used to collect information about network status at different time scales: link utilization, radio status, cache status etc. vICN thus integrates all information about the deployment in a consistent and query-able representation, building on the model introduced in Section 4.3. In the same way vICN provides an API to navigate through structured logs that may assist the whole process of software development.

4.6 An Intent-Centric network management protocol

Current intent-based frameworks (such as OpenStack Heat [217] or the Open-Networking Foundation Boulder project [218]) act as centralized human-interfaces to translate intent into orchestration actions (e.g., configuration files or command-line instructions), which are then offloaded to other modules of the orchestrator. Compared to these, vICN already takes a step further by using a unified intent-based model (Section 4.3.2) for intent specification, resolution, and scheduling but remains nonetheless a single centralized framework. While such centralized approaches represent the first step towards intent-based networking, they suffer from stark limitations. Mainly, they keep orchestration centralized, which results not only in scalability issues (OpenStack, for instance, has over 9M lines of code and running a controller requires at least 3 physical machines with 12 CPU and 64GB of RAM [219]) but also restricts the possibilities offered by intent-based networking in terms of network automation and innovation.

We argue that many limitations of current orchestrators come from the necessity to perform an early and centralized binding from the user's request to a set of device configurations, requiring full knowledge and preventing any further actors to participate in the resolution of the intent. We thus propose to push the intent deeper into the network fabric rather than limiting it to the edge, which means both transporting and processing it in-between network elements or orchestrators. In that regard, the recently established consensus around network management protocols such as NETCONF [220] and YANG [205] is of particular interest.

4.6.1 Intent-based network model

As described in Section 4.3.2, the YANG data model is not too dissimilar to the vICN intent model. In particular, it is extensible and allows decoupling

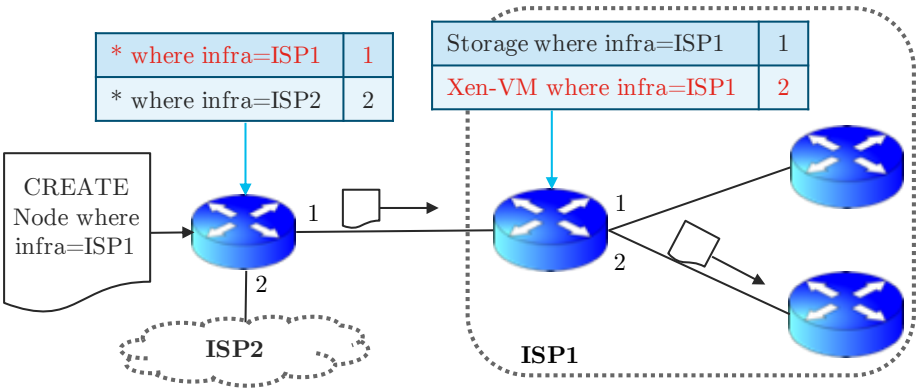


Figure 4.10 – Example of request forwarding

the model structure from its semantic aspects. However, despite sharing some similarities with object-oriented programming or relational databases, YANG does not include all of their features and thus lacks some expressiveness for encoding user intent or exposing relations and constraints within or in-between resources. The use of YANG as the central communication model thus forces intent to be entirely resolved in the origin orchestrator and largely limits the possibility of doing multi-party inference. Furthermore, it forces the orchestrator to keep a complete, fine-grained, and up-to-date database of all the objects present in the network to perform inference (e.g., in the vICN Resource processor – Section 4.3.3). This raises scalability, consistency, and concurrency issues, as multiple modules might modify the same resources synchronously.

To enable distributed intent resolution, a natural solution is thus to extend the reach of the vICN intent model from the orchestrator to the network. Note that, as noted in Section 4.3.2, small tweaks to YANG would effectively realize such a model:

— *Abstraction:* a proposal for extending YANG to support abstract object was introduced in [208], using the `extends` keyword to provide resource inheritance.

— *Foreign models:* Foreign objects could be defined in YANG model by using a `foreign` key, which would make the end-device aware that the object is implemented elsewhere. Foreign objects can then naturally be referenced using `identityref` or `leafref`.

— *Scheduling information:* To provide scheduling information, YANG `groupings` can be completed with similarly defined `sequences`. In this case, `grouping` now indicates that both models can be applied in parallel whereas `sequence` indicates that they must be applied sequentially.

4.6.2 Model-based routing and forwarding

In a second step, we propose to depart from the traditional star deployment where all resources are reachable from a single orchestrator, to a more distributed approach where resources attached to different orchestrators advertise their capabilities throughout the network and allow users' intents to be routed back to those able to best satisfy the request. This essentially means building a routing

and forwarding plane, similar to ICN in that forwarding is done based on an abstract resource rather than a physical network location. Like ICN, our proposal uses location-independent identifiers but these are multidimensional structures representative of the intent itself rather than one-dimensional names. Each orchestrator now becomes a router and is, in addition, able to split (composition) or transform the incoming request (specialization, attribute binding), which gets fully resolved upon reaching end devices of interest. This approach replaces the point-to-point transport of NETCONF by a multipoint-to-multipoint network propagation. An example of such propagation is provided in Figure 4.10.

The design of our intent router closely follows the structure of an ICN router as presented in Section 1.2.1, where the three main data structures have been adapted to our new addressing scheme: the Forwarding Information Base (FIB), used to match incoming intent requests to network locations, the Pending *Intent* Table (PIT), that keeps state to symmetrically route answers back to the origin of the corresponding intent, and a Content Store (CS) that can be used to cache answers.

The FIB is the main component of our router. It maps received resource advertisements to their ingress interface and uses this information to forward an intent to one or several next hops able to further satisfy it. This happens after an eventual local intent-resolution where the intent can be (partially) bound, specialized, or decomposed into multiple sub-intents using the network model. To perform the matching process over multi-dimensional intent objects, we developed a preliminary solution based on maximal subset matching algorithm [221].

Like ICN, we store incoming requests into the PIT along with their ingress interface and associated outgoing sub-requests. The PIT thus collects all the necessary information to send the corresponding answers back up to their origin, thereby implementing symmetric routing. It serves as a distributed scheduling module, keeping track of requests in progress and allowing hop-by-hop reconciliation of concurrent messages. The PIT is thus crucial to ensure the synchronization and consistency of the execution in a multipoint-to-multipoint concurrent environment. Furthermore, the PIT contributes to the scalability of the system by aggregating redundant requests and execute them only once.

The CS caches responses flowing back through the router, and stores the current state of a network model deployment for improved performance. It can be used to access such information with lower RTT and network overhead. It also brings resiliency as the network can still at least partially operate during disconnection periods.

4.7 Summary and future work

In this chapter, we introduce *vICN (virtualized ICN)*, an intent-based unified framework for ICN network configuration, management, and control to complement existing tools, especially for large scale and operational networks deployment. vICN is an object-oriented programming framework rooted in recent advances in SDN/NFV research that provides higher flexibility than existing virtualization solutions. It is specifically tailored to ICN, but its modular design allows for extensions to other technologies. While most of the current software is developed in silos, with significant limitations in terms of optim-

ization, vICN offers the capability to optimize each component of the virtual network to provide carrier-grade service guarantees in terms of programmability, scalability, and reliability. Finally, we present opportunities for further optimization and decentralization of the management process by introducing an intent-centric network management protocol based on the data model that is central to vICN.

There are still many research issues to investigate to achieve intent-based networking. The first and obvious one is the evaluation of the scalability of our system in terms of computing overhead. Indeed, the forwarding algorithms mentioned in Section 4.6.2 are more complex than traditional name-based forwarding in ICN. We have explored preliminary ideas, using work on multi-dimensional forwarding in ICN through maximal subset matching [222]. Our first implementation suggests that it is a viable approach but we leave its description (and - especially - its evaluation) to future work.

The second issue is routing. Indeed, creating a scalable forwarding plane for intent requires to be able to fill up the routers FIB with the corresponding object models. As such, exploring seminal work for routing database queries [223, 224] might give insight into realizing the object-based routing plane. Another approach would be to divide the properties of an object between "*locator*" properties for inter-domain routing and "*identifier*" for intra-domain inference and scheduling. Both approaches should be explored and evaluated in terms of deployment simplicity, expressiveness and scalability.

Finally, another stringent issue of our approach is security. Indeed, in NETCONF the security model resides on securing the point-to-point communication channel (e.g., with SSH or TLS). Our approach currently does not include such a security model. In fact, we need to explore how to authenticate each of the modules that performed inference on the original intent. This is a difficult task that requires a thorough investigation, as a failure to find a scalable and tractable security model would prove fatal for any orchestration protocol.

Chapter 5

Conclusion

ICN is a promising solution for the IoT. Its salient features such as name-based forwarding, native multicast, or object-based security are natural solutions to many of the challenges that the IoT has raised for networks. However, to transform that envisioned fit into actual ICN-IoT deployments, concrete solutions for the network layer in the ICN-enabled IoT must be developed. Indeed, while much of the literature has addressed how ICN can benefit IoT applications, there lacks consistent work on routing and forwarding frameworks for ICN in the IoT context.

In this thesis, three conjoined problems were tackled: (i) how to perform efficiently routing and forwarding in ICN-IoT networks, (ii) how to use ICN forwarding to help IoT applications reach QoS targets, and (iii) how to deploy, manage, and orchestrate edge IoT networks and applications at scale. Overall, these three contributions pave the way for the ICN-enabled IoT. Indeed, they address (i) the problem of setting up a routing plane in WSNs and of (iii) managing the applications and forwarding plane in the edge. Finally, (ii) shows that clever forwarding strategies can help the deployed IoT applications reach their QoS target.

5.1 Geographic routing for the ICN-enabled WSN

In a first step, geographic routing and forwarding are evaluated as a solution for ICN-enabled WSNs. A secure geographic routing framework is proposed, composed of a neighbour authentication protocol, a beaconing protocol, a naming strategy, and a forwarding strategy. For the neighbour authentication protocol, two alternatives are evaluated with similar security features but based respectively on symmetric and asymmetric cryptography, showing that while asymmetric cryptography allows for reducing the number of messages and keeping the transmission local, it has a higher energy cost because of its computational complexity. An analytical model is used to compare geographic routing to an enhanced version of the "flood-and-learn" approaches of the literature in terms of implementability (i.e., CPU and memory footprint on the IoT device) and energy efficiency (i.e., energy overhead caused by the routing protocol). To provide a realistic evaluation, the model is parameterized with data issued from simulation, from previous work, and from experience on a standard IoT sensor

platform. Geographic routing is shown to require less memory than flood-and-learn while also providing large energy savings in cases where the topology is dynamic.

5.2 Popularity-based latency control for Fog applications

In a second step, we look at how clever forwarding can help IoT applications fulfil their QoS requirements. In particular, a popularity-based approach for controlling admission in a limited capacity Fog platform is proposed. Called the LRU-AC, it relies on an LRU meta-cache to estimate the popularity of a request and accept the popular ones in the Fog platform. Using an analytical queueing model, the LRU-AC is shown to provide a good balance between efficiency and practicality compared to either request-blind or oracle-based approaches. An implementation of the LRU-AC on FPGA-hardware using Ageing Bloom filters (ABF) is proposed so as to realize the LRU-AC with minimal latency overhead and maximal throughput. The validity of the implementation is justified using an analytical model and verified via simulation. The ABF-AC is shown to achieve high throughput, low latency overhead, and to multiply the Fog admission-rate by 10 compared to request-blind admission control in the tested scenario.

5.3 Intent-based management of ICN

Finally, a solution for the conjoint deployment, configuration, and management of IoT networks and applications is proposed. Virtualized ICN (vICN), an intent-based unifying framework for network configuration, management, and application orchestration is introduced. It exploits recent progress in Intent-Based Networking to define a new intent-based resource model used to unify network management and application orchestration. Through concrete examples, the programmability, scalability, and reliability of vICN are illustrated. Looking forward, we show how the vICN resource model can actually be used to decentralize intent-based frameworks. In combination with an Intent-Centric protocol, inspired from ICN to perform routing based on intent objects, it sets the foundations for truly developing intent-based networks (instead of intent-based orchestrators) able to automate themselves using intent as the unifying language.

5.4 Future research directions

The work in this thesis relies often on analytical model and simulation, especially in Chapter 2 and Chapter 3. Such an approach offers the advantages of generality, flexibility, and allows to derive insights that are not bound to a given scenario or implementation (which was a goal of this thesis). It should, however, be completed with a system approach, that uses deployment on actual physical hardware to test, e.g., the implementability and scalability of the solutions. It is with that in mind that code-source has been released to implement most

of the schemes proposed in this thesis. Our GPSR module [133] could, for instance, be integrated into a standard ICN-WSN stack (e.g., CCN-Lite [57] over RIOT-OS [58]) and deployed over a public IoT test-bed such as the FIT IoT-LAB [225] to verify, e.g., the energy consumption predicted by the model or to compare the GPSR algorithm with other approaches, such as hyperbolic geographic routing [226, 227].

However, rather than testing each contribution individually on specialized test-beds, vICN can be used to connect them and test their scalability and interoperability. In particular, one could imagine the following deployment:

— a virtualized WSN created thanks to (i) containerized RIOT-OS nodes built in the *native* mode (which allows building RIOT stacks as Linux processes); (ii) virtualized IEEE 802.15.4 links thanks to the `lr-wpan` module of ns-3 [210] integrated into vICN similarly to IEEE 802.11 or LTE;

— a virtualized edge network virtualized with vICN, e.g., using a topology from [228] and assigning realistic computing and storage capacities to each node;

— an IoT application running on these nodes and in a Cloud platform, using data retrieved from the virtualized WSN and deployed via vICN;

— realistic traffic models for either endogenous or exogenous requests.

Deploying such a testbed would allow assessing benefits of our mechanisms in realistic settings across the full IoT vertical, but also to evaluate the opportunities to further development, optimization and integration of the proposed solutions.

For instance, in a submitted patent application [1], a solution is discussed to automate the parameterization of the LRU-AC, which currently depends on the request arrival rate and the popularity distribution. Our proposal exploits a PI-controller to control the response time of the Fog node via the filter-size k_{LRU}. Another research direction would be to extend the Fog admission control to clusters of servers, as the Fog is usually considered as highly distributed [17, 48]. While independent LRU-AC modules could be deployed independently in front of each Fog node, this might result in an under-optimized utilisation of the global storage and computing resources. On the other hand, enforcing collaboration between caches can be difficult because, e.g., of the time scale of the feedback loop versus the request arrival rate, or because of the difficulty of reaching distributed consensus. A candidate solution should thus not only have strong mathematical guarantees in terms of resource utilization but also be flexible and adaptive enough to fit the IoT context. The LRU-AC, combined, for instance, with a clever request-aware load-balancing technique, could be the foundation for such a distributed QoS control framework.

1. M. Enguehard, Y. Desmouceaux, P. Pfister, M. Townsley, E. Vyncke. EFFICIENT AND FLEXIBLE LOAD-BALANCING FOR CLUSTERS OF CACHES UNDER LATENCY CONSTRAINT. USPTO Application Number 16/261,462

Bibliography

[1] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, may 1974.

[2] J. Martocci, P. D. Mil, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks," Internet Requests for Comments, RFC Editor, RFC 5867, Jun. 2010.

[3] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication systems for building automation and control," *Proc. IEEE*, vol. 93, no. 6, pp. 1178–1203, jun 2005.

[4] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *IEEE Computer*, vol. 46, no. 7, pp. 62–69, Jul. 2013.

[5] A. Brandt, J. Buron, and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks," Internet Requests for Comments, RFC Editor, RFC 5826, Apr. 2010.

[6] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A zigbee-based home automation system," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 422–430, may 2009.

[7] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," Internet Requests for Comments, RFC Editor, RFC 5673, Oct. 2009.

[8] F. De Pellegrini, D. Miorandi, S. Vitturi, and A. Zanella, "On the use of wireless networks at low level of factory automation systems," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 2, pp. 129–143, may 2006.

[9] S. hai An, B.-H. Lee, and D.-R. Shin, "A survey of intelligent transportation systems," in *Proc. 3rd International Conference on Computational Intelligence, Communication Systems and Networks*. IEEE, jul 2011.

[10] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proc. IEEE*, vol. 98, no. 11, pp. 1903–1917, nov 2010.

[11] A. Cocchia, "Smart and digital city: A systematic literature review," in *Smart city*. Springer, 2014, pp. 13–43.

[12] M. Dohler, T. Watteyne, T. Winter, and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy networks," Internet Requests for Comments, RFC Editor, RFC 5548, May 2009.

[13] J.-F. Balcon, "Expérimentation: Cisco mesure la fréquentation de la place de la Nation," http://gblogs.cisco.com/fr-smartcities/2016/03/31/experimentation-cisco-mesure-la-frequentation-de-la-place-de-la-nation/, Mar. 2016, consulted in Jan 2017.

[14] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid — the new and improved power grid: A survey," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 944–980, 2012.

[15] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a mobile sensor," *Procedia Computer Science*, vol. 34, pp. 286–295, 2014.

[16] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, aug 2008.

[17] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st Edition of the Workshop on Mobile Cloud Computing (MCC'12)*, 2012.

[18] J. Höller, D. Boyle, S. Karnouskos, S. Avesand, C. Mulligan, and V. Tsiatsis, *From machine-to-machine to the internet of things*. Elsevier, 2014.

[19] Cisco Virtual Networking Index, "The zettabyte era: Trends and analysis," 2017, [Accessed 2018/06/12]. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf

[20] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-s – a publish/subscribe protocol for wireless sensor networks," in *Proc. 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, jan 2008.

[21] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, and M. Mohammadi, "Toward better horizontal integration among IoT services," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 72–79, sep 2015.

[22] R. Silva, J. S. Silva, M. Simek, and F. Boavida, "Why should multicast be used in WSNs," in *Proc. 2008 IEEE International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2008.

[23] W. Shang, Y. Yu, R. Droms, and L. Zhang, "Challenges in IoT networking via TCP/IP architecture," Named-Data Networking Project, techreport NDN-0038, Feb. 2016.

[24] 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 *Proc. 1st international conference on Information-centric networking - ICN'14*. New York, NY, USA: ACM, 2014, pp. 77–86.

[25] G. Montenegro, "Transmission of IPv6 packets over IEEE 802.15.4 networks," RFC 4944, Sep. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4944.txt

[26] T. Watteyne, T. Winter, D. Barthel, and M. Dohler, "Routing Requirements for Urban Low-Power and Lossy Networks," RFC 5548, May 2009. [Online]. Available: https://rfc-editor.org/rfc/rfc5548.txt

[27] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347, Jan. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6347.txt

[28] 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

[29] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL)," in *Proc. 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, oct 2011.

[30] S. Cespedes, X. Shen, and C. Lazo, "IP mobility management for vehicular communication networks: challenges and solutions," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 187–194, may 2011.

[31] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an analysis of security issues, challenges, and open problems in the internet of things," in *Proc. 2015 IEEE World Congress on Services*. IEEE, jun 2015.

[32] M. Vucinic, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "OSCAR: Object security architecture for the internet of things," in *Proc. 2014 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, jun 2014.

[33] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IoT gateway: Bridging wireless sensor networks into internet of things," in *Proc. 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*. IEEE, dec 2010.

[34] R. B. Miller, "Response time in man-computer conversational transactions," in *Proc. International Workshop on Managing Requirements Knowledge (AFIPS)*. ACM, Dec. 1968, pp. 267–277.

[35] 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, Mar. 2016.

[36] R. Ravindran, Y. Zhang, L. A. Grieco, A. Lindgren, J. Burke, B. Ahlgren, and A. Azgin, "Design Considerations for Applying ICN to IoT," Internet Engineering Task Force, Internet-Draft draft-irtf-icnrg-icniot-02, Oct. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-icniot-02

[37] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th international conference on Emerging networking experiments and technologies - CoNEXT'09*. ACM, 2009.

[38] L. Pouzin, "Presentation and major design aspects of the CYCLADES computer network," in *Proc. 3rd ACM symposium on Data communications and Data networks Analysis and design - DATACOMM'73*. ACM, 1973.

[39] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, p. 2, dec 2012.

[40] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking,"

ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 66–73, jul 2014.

[41] The Linux Foundation, "FD.io CICN project," 2017. [Online]. Available: https://wiki.fd.io/view/Cicn

[42] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (NetInf) – an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, apr 2013.

[43] M. Mosko, I. Solis, and C. A. Wood, "CCNx Semantics," Internet Engineering Task Force, Internet-Draft draft-irtf-icnrg-ccnxsemantics-10, Jan. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-ccnxsemantics-10

[44] ——, "CCNx Messages in TLV Format," Internet Engineering Task Force, Internet-Draft draft-irtf-icnrg-ccnxmessages-09, Jan. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-ccnxmessages-09

[45] B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. B. Davies, A. Molinaro, and S. Eum, "Information-Centric Networking: Baseline Scenarios," RFC 7476, Mar. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7476.txt

[46] M. Amadeo, C. Campolo, A. Molinaro, and G. Ruggeri, "Content-centric wireless networking: A survey," *Computer Networks*, vol. 72, pp. 1–13, oct 2014.

[47] S. Cirani, L. Davoli, G. Ferrari, R. Leone, P. Medagliani, M. Picone, and L. Veltri, "A scalable and self-configuring architecture for service discovery in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 508–521, oct 2014.

[48] J. A. Khan, C. Westphal, and Y. Ghamri-Doudane, "A content-based centrality metric for collaborative caching in information-centric fogs," in *Proc. 2017 IFIP Networking Conference and Workshops*. IEEE, Jun. 2017.

[49] O. Hahm, E. Baccelli, T. C. Schmidt, M. Wählisch, C. Adjih, and L. Massoulié, "Low-power internet of things with NDN & cooperative caching," in *Proc. 4th ACM Conference on Information-Centric Networking - ICN'17*. ACM, 2017.

[50] K. S. Prabh and T. F. Abdelzaher, "Energy-conserving data cache placement in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 178–203, nov 2005.

[51] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking based internet of things (ICN-IoT)," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[52] I. Stojmenović and S. Olariu, "Data-centric protocols for wireless sensor networks," in *Handbook of Sensor Networks*. John Wiley & Sons, Inc., sep 2005, pp. 417–456.

[53] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in

Proc. 6th annual international conference on Mobile computing and networking - MobiCom'00. New York, NY, USA: ACM, 2000, pp. 56–67.

[54] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton, "Named data networking: A natural design for data collection in wireless sensor networks," in *Proc. 2013 IFIP Wireless Days*. IEEE, nov 2013.

[55] O. Ascigil, S. Reñé, G. Xylomenos, I. Psaras, and G. Pavlou, "A keyword-based ICN-IoT platform," in *Proc. 4th ACM Conference on Information-Centric Networking - ICN'17*. ACM, 2017.

[56] Y. Abidy, B. Saadallahy, A. Lahmadi, and O. Festor, "Named data aggregation in wireless sensor networks," in *Proc. IEEE Network Operations and Management Symposium (NOMS)*. IEEE, May 2014.

[57] "CCN Lite: Lightweight implementation of the content centric networking protocol," http://www.ccn-lite.net.

[58] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. Schmidt, "RIOT OS: Towards an OS for the internet of things," in *Proc. 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, Apr. 2013.

[59] C. Gündogan, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt, and M. Wählisch, "NDN, CoAP, and MQTT: A comparative measurement study in the IoT," in *Proc. 5th ACM Conference on Information-Centric Networking (ICN '18)*. ACM, Sep. 2018.

[60] A. Banks and R. Gupta, "Mqtt version 3.1.1," OASIS standard, 2014.

[61] N. Fotiou, H. Islam, D. Lagutin, T. Hakala, and G. C. Polyzos, "CoAP over ICN," in *Proc. 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, Nov. 2016, pp. 1–4.

[62] W. Shang, A. Afanasyev, and L. Zhang, "The design and implementation of the NDN protocol stack for RIOT-OS," in *Proc. 2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, dec 2016.

[63] 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 *Proc. 1st International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, apr 2016.

[64] Z. Ren, M. A. Hail, and H. Hellbruck, "CCN-WSN - a lightweight, flexible content-centric networking protocol for wireless sensor networks," in *Proc. IEEE 18th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'13)*. IEEE, Apr. 2013, pp. 123–128.

[65] J. Pfender, A. Valera, and W. K. Seah, "Performance comparison of caching strategies for information-centric IoT," in *Proc. 5th ACM Conference on Information-Centric Networking - ICN'18*, 2018.

[66] L. M. J.S.M., V. Lokesh, and G. C. Polyzos, "Energy efficient context based forwarding strategy in named data networking of things," in *Proc. 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16*. ACM, 2016.

[67] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Secure sensing over named data networking," in *Proc. 13th International Symposium on Network Computing and Applications*. IEEE, Aug. 2014, pp. 175–180.

[68] A. Compagno, M. Conti, and R. Droms, "OnboardICNg: a secure protocol for on-boarding IoT devices in ICN," in *Proc. 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN'16*. New York, NY, USA: ACM, Sep. 2016, pp. 166–175.

[69] T. Mick, R. Tourani, and S. Misra, "LASeR: Lightweight authentication and secured routing for NDN IoT in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 755–764, apr 2018.

[70] J. A. Khan, C. Westphal, and Y. Ghamri-Doudane, "A popularity-aware centrality metric for content placement in information centric networks," in *Proc. 2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, mar 2018.

[71] M. Wang, J. Wu, G. Li, J. Li, and Q. Li, "Fog computing based content-aware taxonomy for caching optimization in information-centric networks," in *Proc. 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, may 2017.

[72] M. Sifalakis, B. Kohler, C. Christopher, and C. Tschudin, "An information centric network for computing the distribution of computations," in *Proc. 1st international conference on Information-centric networking - ICN '14*. ACM, 2014.

[73] M. J. Fischer, "Lambda-calculus schemata," *LISP and Symbolic Computation*, vol. 6, no. 3-4, pp. 259–287, nov 1993.

[74] M. Król and I. Psaras, "NFaaS: : named function as a service," in *Proc. 4th ACM Conference on Information-Centric Networking - ICN'17*. ACM, 2017.

[75] C. Scherb, D. Grewe, M. Wagner, and C. Tschudin, "Resolution strategies for networking the IoT at the edge via named functions," in *Proc. 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, jan 2018.

[76] W. Shang, Z. Wang, A. Afanasyev, J. Burke, and L. Zhang, "Breaking out of the cloud: Local trust management and rendezvous in named data networking of things," in *Proc. 2nd International Conference on Internet-of-Things Design and Implementation - IoTDI'17*. IEEE, 2017.

[77] S. S. Adhatarao, M. Arumaithurai, and X. Fu, "FOGG: A fog computing based gateway to integrate sensor networks to internet," in *Proc. 29th International Teletraffic Congress (ITC 29)*. IEEE, sep 2017.

[78] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, "FCSS: Fog computing based content-aware filtering for security services in information centric social networks," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2017.

[79] D. Nguyen, Z. Shen, J. Jin, and A. Tagami, "ICN-fog: An information-centric fog-to-fog architecture for data communications," in *Proc. 2017 IEEE Global Communications Conference (GLOBECOM 2017)*. IEEE, dec 2017.

[80] M. Chen, "NDNC-BAN: Supporting rich media healthcare services via named data networking in cloud-assisted wireless body area networks," *Information Sciences*, vol. 284, pp. 142–156, Nov. 2014.

[81] H. Zhang, Z. Wang, C. Scherb, C. Marxer, J. Burke, L. Zhang, and C. Tschudin, "Sharing mHealth data via named data networking," in *Proc. 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16*. ACM, 2016.

[82] D. Saxena and V. Raychoudhury, "Design and verification of an NDN-based safety-critical application: A case study with smart healthcare," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–15, 2017.

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

[84] U. D. Silva, A. Lertsinsrubtavee, A. Sathiaseelan, and K. Kanchanasut, "Named data networking based smart home lighting," in *Proc. ACM Conference on Special Interest Group on Data Communication - - SIGCOMM '16*. ACM, 2016.

[85] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using named data networking," *IEEE Network*, vol. 28, no. 3, pp. 50–56, May 2014.

[86] G. Piro, I. Cianci, L. A. Grieco, G. Boggia, and G. Camarda, "Information centric services in smart cities," *Journal of Systems and Software*, vol. 88, pp. 169–188, Feb. 2014.

[87] 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.

[88] S. H. Bouk, S. H. Ahmed, D. Kim, and H. Song, "Named-data-networking-based ITS for smart cities," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 105–111, jan 2017.

[89] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: a survey and future perspectives," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 98–104, Feb. 2016.

[90] 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.

[91] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest forwarding by geolocations in vehicular named data networking," in *Proc. 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. Boston, MA, USA: IEEE, jun 2015.

[92] P. H. Stott, "The UTM grid reference system." *IA. The Journal of the Society for Industrial Archeology*, vol. 3, no. 1, pp. 1–14, 1977.

[93] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang, "Scaling NDN routing: Old tale, new design," NDN Project, techreport NDN-0004, 2013.

[94] H. Yuan, T. Song, and P. Crowley, "Scalable NDN forwarding: Concepts, issues and principles," in *Proc. 21st International Conference on Computer Communications and Networks (ICCCN)*. IEEE, jul 2012.

[95] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, apr 2013.

[96] H. B. Abraham and P. Crowley, "Forwarding strategies for applications in named data networking," in *Proc. 2016 Symposium on Architectures for Networking and Communications Systems - ANCS'16*. ACM, 2016.

[97] M. Enguehard, Y. Desmouceaux, and G. Carofiglio, "Efficient latency control in Fog deployments via hardware-accelerated popularity estimation," 2019, under review.

[98] M. Enguehard, R. E. Droms, and D. Rossi, "On the cost of geographic forwarding for information-centric things," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 1150–1163, Dec. 2018.

[99] Y. Desmouceaux, M. Enguehard, V. Nguyen, P. Pfister, W. Shao, and É. Vyncke, "A content-aware data-plane for scalable video delivery," in *Proc. 16th IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019.

[100] M. Enguehard, G. Carofiglio, and D. Rossi, "A popularity-based approach for effective cloud offload in fog deployments," in *Proc. 30th International Teletraffic Congress (ITC 30)*. Vienna, Austria: IEEE, Sep. 2018.

[101] M. Sardara, L. Muscariello, J. Augé, M. Enguehard, A. Compagno, and G. Carofiglio, "Virtualized ICN (vICN): towards a unified network virtualization framework for ICN experimentation," in *Proc. 4th ACM Conference on Information-Centric Networking - ICN'17*. ACM, 2017, pp. 109–115.

[102] M. Enguehard, R. Droms, and D. Rossi, "SLICT: Secure localized information centric things," in *Proc. 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16 (IC5G workshop)*. New York, NY, USA: ACM, 2016, pp. 255–260.

[103] ——, "Poster: On the cost of secure association of information centric things," in *Proc. 2016 conference on 3rd ACM Conference on Information-Centric Networking - ICN'16*. New York, NY, USA: ACM, 2016, pp. 207–208.

[104] J. Augé, G. Carofiglio, M. Enguehard, L. Muscariello, and M. Sardara, "Simple and efficient ICN network virtualization with vICN," in *Proc. 4th ACM Conference on Information-Centric Networking - ICN '17*. ACM, 2017.

[105] J. Augé and M. Enguehard, "A network protocol for distributed orchestration using intent-based forwarding," in *Proc. 16th IFIP/IEEE International Symposium on Integrated Network Management (IM) - Demonstration Session*, Apr. 2019.

[106] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. 5th annual ACM/IEEE international conference on Mobile computing and networking - MobiCom '99*. ACM, 1999.

[107] D. Chen and P. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 1, pp. 50–67, 2007.

[108] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. 6th annual international conference on Mobile computing and networking - MobiCom'00*. New York, NY, USA: ACM, 2000, pp. 243–254.

[109] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 720–734, Oct. 2016.

[110] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proc. 2nd international conference on Embedded networked sensor systems - SenSys'04*. New York, NY, USA: ACM, 2004, pp. 214–226.

[111] A. Rowe, M. E. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. F. Moura, and L. Soibelman, "Sensor andrew: Large-scale campus-wide sensing and actuation," *IBM Journal of Research and Development*, vol. 55, no. 1.2, pp. 6:1–6:14, jan 2011.

[112] M. Amadeo, O. Briante, C. Campolo, A. Molinaro, and G. Ruggeri, "Information-centric networking for m2m communications: Design and deployment," *Computer Communications*, vol. 89-90, pp. 105–116, Sep. 2016.

[113] Mairie de Paris, "Paris data," https://opendata.paris.fr.

[114] H. Tschofenig and M. Pegourie-Gonnard, "Performance of state-of-the-art cryptographyon ARM-based microprocessors," NIST Lightweight Cryptography Workshop 2015, Jul. 2015.

[115] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5246.txt

[116] G. de Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *Proc. 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, oct 2008, pp. 580–585.

[117] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Computer Networks*, vol. 54, no. 17, pp. 2967–2978, dec 2010.

[118] H. Shafagh, A. Hithnawi, A. Droescher, S. Duquennoy, and W. Hu, "Talos: Encrypted query processing for the Internet of Things," in *Proc. 13th ACM Conference on Embedded Networked Sensor Systems - SenSys'15*. New York, NY, USA: ACM, 2015, pp. 197–210.

[119] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proc. ACM SIGCOMM workshop on Information-centric networking - ICN'11*. New York, NY, USA: ACM, 2011, pp. 13–18.

[120] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, p. 62, jun 2012.

[121] G. Carofiglio, M. Gallo, and L. Muscariello, "Optimal multipath congestion control and request forwarding in information-centric networks:

Protocol design and experimentation," *Computer Networks*, vol. 110, pp. 104–117, dec 2016.

[122] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named data networking," in *22nd International Conference on Computer Communication and Networks (ICCCN)*. IEEE, jul 2013.

[123] W. A. Simpson, D. T. Narten, E. Nordmark, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861, Sep. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4861.txt

[124] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[125] M. Amadeo, C. Campolo, and A. Molinaro, "Multi-source data retrieval in IoT via named data networking," in *Proc. 1st international conference on Information-centric networking - ICN '14*. New York, NY, USA: ACM, 2014, pp. 67–76.

[126] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *Proc. 22nd annual symposium on Principles of distributed computing - PODC'03*. New York, NY, USA: ACM, 2003, pp. 63–72.

[127] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli, "BLR: beacon-less routing algorithm for mobile ad hoc networks," *Computer Communications*, vol. 27, no. 11, pp. 1076–1086, jul 2004, applications and Services in Wireless Networks.

[128] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz, "BOSS: Beacon-less on demand strategy for geographic routing in wireless sensor networks," in *Proc. 2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MobHoc)*. IEEE, oct 2007, pp. 1–10.

[129] L. Wang, O. Waltari, and J. Kangasharju, "MobiCCN: Mobility support with greedy routing in content-centric networks," in *Proc. 2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, dec 2013, pp. 2069–2075.

[130] D. Pesavento, G. Grassi, C. E. Palazzi, and G. Pau, "A naming scheme to represent geographic areas in NDN," in *Proc. 2013 IFIP Wireless Days*, Nov. 2013, pp. 1–3.

[131] H. Ma, L. Liu, A. Zhou, and D. Zhao, "On networking of internet of things: Explorations and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 441–452, Aug. 2016.

[132] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annual IEEE International Conference on Local Computer Networks*. IEEE (Comput. Soc.), Nov. 2004, pp. 455–462.

[133] M. Enguehard, "marceleng/geographic-icthings: Code for the paper: On the cost of geographic forwarding for information centric things," https://github.com/marceleng/geographic-icthings, Apr. 2018.

[134] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless internet of things," in *Proc. 2015 International Conference on Recent Advances in Internet of Things (RIoT)*. IEEE, apr 2015, pp. 1–6.

[135] C. Anastasiades, J. Weber, and T. Braun, "Dynamic unicast: Information-centric multi-hop routing for mobile ad-hoc networks," *Computer Networks*, vol. 107, pp. 208–219, oct 2016.

[136] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Named data networking for IoT: An architectural perspective," in *Proc. 2014 European Conference on Networks and Communications (EuCNC)*. IEEE, Jun. 2014.

[137] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proc. 35th Annual Hawaii International Conference on System Sciences*, Jan. 2002, pp. 3866–3875.

[138] *CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee® Applications*, Texas Instrument, Dec. 2012, revised April 2015.

[139] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "OpenMote: Open-source prototyping platform for the industrial IoT," in *Proc. International Conference on Ad Hoc Networks*. Springer, 2015, pp. 211–222.

[140] M. Mosko and C. Tschudin, "CCN and NDN TLV encodings in 802.15.4 packets," https://www.ietf.org/mail-archive/web/icnrg/current/pdfs9ieLPWcJI.pdf, Jan. 2015, consulted on March 17, 2017.

[141] PARC, "The CCNx project," https://blogs.parc.com/ccnx/.

[142] O. Hahm, E. Baccelli, T. C. Schmidt, M. Wahlisch, and C. Adjih, "A named data network approach to energy efficiency in IoT," in *Proc. 2016 IEEE Globecom Workshops*. IEEE, dec 2016.

[143] K. Roussel, Y.-Q. Song, and O. Zendra, "Using Cooja for WSN simulations: Some new uses and limits," in *Proc. 2016 International Conference on Embedded Wireless Systems and Networks (EWSN'16)*. USA: Junction Publishing, 2016, pp. 319–324.

[144] S. Pollin, M. Ergen, S. C. Ergen, B. Bougard, L. V. D. Perre, I. Moerman, A. Bahai, P. Varaiya, and F. Catthoor, "Performance analysis of slotted carrier sense ieee 802.15.4 medium access layer," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3359–3371, Sep. 2008.

[145] K. Nisimova, "Energy of a 1.5 V battery," http://hypertextbook.com/facts/2001/KhalidaNisimova.shtml, consulted on 20 April 2017.

[146] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proc. 2nd ACM SIGCOMM workshop on Mobile cloud computing - MCC'13*. ACM, 2013.

[147] Z. Chen, R. Klatzky, D. Siewiorek, M. Satyanarayanan, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, and P. Pillai, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. 2nd ACM/IEEE Symposium on Edge Computing (SEC'17)*. ACM, 2017.

[148] "Aws greengrass," https://aws.amazon.com/greengrass.

[149] Y. Niu, F. Liu, X. Fei, and B. Li, "Handling flash deals with soft guarantee in hybrid cloud," in *Proc. 2017 IEEE Conference on Computer Communications (INFOCOM 2017)*. IEEE, may 2017.

[150] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, apr 2018.

[151] M. Blott, K. Karras, L. Liu, K. A. Vissers, J. Bär, and Z. István, "Achieving 10Gbps line-rate key-value stores with FPGAs." in *Proc. 5th USENIX Workshopon Hot Topics in Cloud Computing (HotCloud'13)*, 2013.

[152] M. Yoon, "Aging bloom filter with two active buffers for dynamic sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 134–138, jan 2010.

[153] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 gbps as research commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, sep 2014.

[154] L. Muscariello, G. Carofiglio, J. Auge, and M. Papalini, "Hybrid Information-Centric Networking," Internet Engineering Task Force, Internet-Draft draft-muscariello-intarea-hicn-01, Dec. 2018, work in Progress.

[155] P. Bosshart, G. Varghese, D. Walker, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, and A. Vahdat, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, jul 2014.

[156] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis - SC'11*. ACM, 2011.

[157] T. Janaszka, D. Bursztynowski, and M. Dzida, "On popularity-based load balancing in content networks," in *Proc. 24th International Teletraffic Congress*, 2012, p. 12.

[158] G. Carofiglio, L. Mekinda, and L. Muscariello, "FOCAL: Forwarding and caching with latency awareness in information-centric networking," in *Proc. 2015 IEEE Globecom Workshops*. IEEE, dec 2015, pp. 1–7.

[159] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *Proc. 18th IEEE Conference on Computer Communications (INFOCOM '99)*, vol. 1. IEEE, 1999.

[160] C. Imbrenda, L. Muscariello, and D. Rossi, "Analyzing cacheable traffic in isp access networks for micro cdn applications via content-centric networking," in *Proc. 1st international conference on Information-centric networking - ICN '14*. ACM, 2014.

[161] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: why it matters and how to model it," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 5, pp. 5–12, nov 2013.

[162] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," *ACM*

SIGMETRICS Performance Evaluation Review, vol. 33, no. 1, p. 291, jun 2005.

[163] M. Nabe, M. Murata, and H. Miyahara, "Analysis and modeling of world wide web traffic for capacity dimensioning of internet access lines," *Performance Evaluation*, vol. 34, no. 4, pp. 249–271, dec 1998.

[164] J. Boyer, F. Guillemin, P. Robert, and B. Zwart, "Heavy tailed m/g/1-PS queues with impatience and admission control in packet networks," in *Proc. 22nd IEEE Conference on Computer Communications (INFOCOM 2003)*, vol. 1. IEEE, 2003.

[165] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, sep 2002.

[166] F. P. Kelly, *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[167] G. F. Newell, "The M/G/∞ queue," *Journal on Applied Mathematic*, vol. 14, no. 1, 1966.

[168] D. Shasha and T. Johnson, "2q: A low overhead high performance buffer management replacement algoritm," in *Proc. 20th International Conference on Very Large Databases*, 1994.

[169] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, no. 3, pp. 1–28, may 2016.

[170] K. Svanberg, "The method of moving asymptotes – a new method for structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 24, no. 2, pp. 359–373, feb 1987.

[171] S. G. Johnson, "The NLopt nonlinear-optimization package," http://ab-initio.mit.edu/nlopt.

[172] M. Enguehard and Y. Desmouceaux, "marceleng/queueing-network-simulator: a simulator for queueing networks," https://github.com/marceleng/queueing-network-simulator, Jan 2019.

[173] J.-L. Brelet, "Using block ram for high performance read/write cams," Xilinx Inc., Application Notes, 2000.

[174] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," *IEEE Micro*, vol. 24, no. 1, pp. 52–61, jan 2004.

[175] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast hash table lookup using extended bloom filter: an aid to network processing," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 181–192, 2005.

[176] H. Song, F. Hao, M. Kodialam, and T. Lakshman, "IPv6 lookups using distributed and load balanced Bloom filters for 100Gbps core router line cards," in *Proc. 28th IEEE Conference on Computer Communications (INFOCOM 2009)*. IEEE, apr 2009, pp. 2518–2526.

[177] M. Yu, A. Fabrikant, and J. Rexford, "BUFFALO: Bloom filter forwarding architecture for large organizations," in *Proc. 5th international conference on Emerging networking experiments and technologies - CoNEXT'09*. ACM, 2009, pp. 313–324.

[178] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. 24th International Teletraffic Congress*, 2012, p. 8.

[179] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, Jul. 2014.

[180] P4.org, "P4 → NetFPGA: A low-cost solution for testing P4 programs in hardware," https://p4.org/p4/p4-netfpga-a-low-cost-solution-for-testing-p4-programs-in-hardware.html, Oct. 2017.

[181] A. Sivaraman, A. Cheung, M. Budiu, C. Kim, M. Alizadeh, H. Balakrishnan, G. Varghese, N. McKeown, and S. Licking, "Packet transactions: High-level programming for line-rate switches," in *Proc. ACM Conference on Special Interest Group on Data Communication - SIGCOMM'16*. ACM, 2016, pp. 15–28.

[182] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen, "A reliable randomized algorithm for the closest-pair problem," *Journal of Algorithms*, vol. 25, no. 1, pp. 19–51, oct 1997.

[183] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing – a key technology towards 5g," ETSI white paper, 2015.

[184] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1786–1794, Sep. 2013.

[185] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, feb 2013.

[186] "The OpenStack foundation," https://www.openstack.org/.

[187] "Kubenetes: Production-grade container orchestration," https://kubernetes.io/.

[188] A. Lerner, J. Skorupa, and S. Ganguli, "Innovation insight: Intent-based networking systems," Gartner, Tech. Rep., 2017.

[189] P. A. Aranda Gutiérrez and D. R. López, "Fighting your way through the jungle of intent," *IEEE Softwarization*, Sep. 2016.

[190] "Networking on a new level," MIT Technology Review, Nov. 2017.

[191] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, jan 2018.

[192] "Future internet research and experimentation," https://www.ict-fire.eu.

[193] C. M. Cabral, C. E. Rothenberg, and M. F. Magalhães, "Mini-CCNx: Fast prototyping for named data networking," in *Proc. 3rd ACM SIGCOMM workshop on Information-centric networking - ICN '13*. ACM, 2013, pp. 33–34.

[194] The Linux Foundation, "Virtualized ICN," https://wiki.fd.io/view/vicn.

[195] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed," *Computer Networks*, vol. 57, no. 16, pp. 3207–3221, Nov. 2013.

[196] R. Ravindran, A. Chakraborti, S. O. Amin, A. Azgin, and G. Wang, "5g-ICN: Delivering ICN services over 5g using network slicing," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 101–107, may 2017.

[197] "Mininet," http://mininet.org/.

[198] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the Emulab network testbed," in *USENIX 2008 Annual Technical Conference (ATC)*. Berkeley, CA, USA: USENIX Association, 2008, pp. 113–128.

[199] M. Lacage, M. Ferrari, M. Hansen, T. Turletti, and W. Dabbous, "NEPI: using independent simulators, emulators, and testbeds for easy experimentation," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 60–65, jan 2010.

[200] The Linux Foundation, "OpenStack Chef," https://wiki.openstack.org/wiki/Chef, 2017.

[201] "Puppet openstack," https://wiki.openstack.org/wiki/Puppet.

[202] E. T. S. Institute, "Network functions virtualisation (NFV); management and orchestration," European Telecommunications Standards Institute (ETSI), Tech. Rep. GS NFV-MAN 001, 2014.

[203] S. Hares, "Intent-Based Nemo Overview," Internet Engineering Task Force, Internet-Draft draft-hares-ibnemo-overview-01, Oct. 2015, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-hares-ibnemo-overview-01

[204] C. Prakash, Y. Zhang, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, and P. Sharma, "PGA: Using graphs to express and automatically reconcile network policies," in *Proc. ACM Conference on Special Interest Group on Data Communication - SIGCOMM'15*. ACM, 2015.

[205] M. Bjorklund, "The YANG 1.1 Data Modeling Language," RFC 7950, Aug. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7950.txt

[206] C. J. Date and H. Darwen, *Foundation for object/relational databases: the third manifesto*. Addison-Wesley Professional, 1998.

[207] J. Paredaens, "On the expressive power of the relational algebra," *Information Processing Letters*, vol. 7, no. 2, pp. 107–111, Feb. 1978.

[208] S. Kuryla, B. Linowski, and M. Ersue, "Extending YANG with Language Abstractions," RFC 6095, Mar. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6095.txt

[209] A. Brown, *The architecture of open source applications (SQLAlchemy)*. Kristian Hermansen, 2012, vol. 2.

[210] "The Network Simulator version 3," https://www.nsnam.org/.

[211] A. K. Mackworth, "The logic of constraint satisfaction," *Artificial Intelligence*, vol. 58, no. 1-3, pp. 3–20, dec 1992.

[212] O. Sinnen, *Task scheduling for parallel systems.* John Wiley & Sons, 2007, vol. 60.

[213] R. Sakellariou and H. Zhao, "A hybrid heuristic for DAG scheduling on heterogeneous systems," in *Proc. 18th International Parallel and Distributed Processing Symposium, 2004.* IEEE, 2004, p. 111.

[214] The Linux Foundation, "Vector Packet Processing - Fast Data I/O," https://wiki.fd.io/view/VPP/.

[215] "Linux containers," https://linuxcontainers.org/.

[216] The Linux Foundation, "Open vSwitch," http://openvswitch.org/, 2017.

[217] "Heat - OpenStack," https://wiki.openstack.org/Heat.

[218] "Boulder - Open Networking Foundation," https://www.opennetworking.org/incubator-projects/boulder/.

[219] R. Kofman, "OpenStack hardware requirements and capacity planning: Servers, CPU and RAM," https://www.stratoscale.com/blog/openstack/openstack-hardware-requirements-and-capacity-planning-servers-cpu-and-ram-part-1/, 2018.

[220] R. Enns, M. Bjorklund, A. Bierman, and J. Schönwälder, "Network Configuration Protocol (NETCONF)," RFC 6241, Jun. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6241.txt

[221] N. Alon and R. Yuster, "Fast algorithms for maximum subset matching and all-pairs shortest paths in graphs with a (not so) small vertex cover," in *Proc. European Symposium on Algorithms (ESA 2007).* Springer Berlin Heidelberg, 2007, pp. 175–186.

[222] D. Rogora, M. Papalini, K. Khazaei, A. Margara, A. Carzaniga, and G. Cugola, "High-throughput subset matching on commodity gpu-based systems," in *Proc. 12th European Conference on Computer Systems (EUROSYS'17).* ACM, 2017, pp. 513–526.

[223] A. Kumar, J. Xu, and E. W. Zegura, "Efficient and scalable query routing for unstructured peer-to-peer networks," in *Proc. 24th IEEE Conference on Computer Communications (INFOCOM 2005)*, vol. 2. IEEE, 2005, pp. 1162–1173.

[224] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica, "Querying the internet with PIER," in *Proc. 29th international conference on very large data bases.* Elsevier, 2003, pp. 321–332.

[225] 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. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT).* IEEE, Dec. 2015.

[226] W. Zeng, R. Sarkar, F. Luo, X. Gu, and J. Gao, "Resilient routing for sensor networks using hyperbolic embedding of universal covering space," in *Proc. 29th IEEE Conference on Computer Communications (INFOCOM 2010).* IEEE, mar 2010, pp. 1–9.

[227] V. Lehman, A. Gawande, B. Zhang, L. Zhang, R. Aldecoa, D. Krioukov, and L. Wang, "An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN," in *Proc. 24th IEEE/ACM International Symposium on Quality of Service (IWQoS).* IEEE, jun 2016.

[228] 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.

[229] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

Appendix A

Appendix of Chapter 3

A.1 Computing the Fog hit rate for the LRU-AC

As explained in Sec. IV.D of [169], our method to compute the cache hit rate in the Fog assumes independence between the arrivals at the filter and at the Fog cache. To remove that assumption, we can consider the Discrete Time Markov Chain (DTMC) presented in Fig. 4 of [169] (and summarized in Figure A.1). It represents the presence of a given piece of content in the filter and in the cache at the time of arrival of a request for said content. In particular, an unwritten assumption of the model in [169] is that the characteristic time t_1 of the filter is smaller than the characteristic time t_2 of the cache ($t_1 < t_2$). Thus, $q_a = 1 - e^{-t_1 q(n)}$ is the probability that the inter-arrival time is smaller than t_1; $q_b = e^{-t_2 q(n)}$ is the probability that the inter-arrival time is bigger than t_2; $q_c = 1 - (q_a + q_b)$ is the probability that the inter-arrival time is bigger than t_1 but smaller than t_2.

We can now predict the Fog hit rate using the balance equations for the given chain. Note that compared to [169], the hit probability is only derived from the probability $p_{(1,1)}$ of being in the state $(1, 1)$ (when the content is in both the filter and the cache), whereas for the LRU-2Q case the state $(0, 1)$ (where the content is only available in the cache) also leads to a hit. In particular, the Fog hit probability for content r reads:

$$h_f(r) = \frac{q_a^2}{q_a + q_b}$$

Furthermore, t_2 can be computed by resolving the following equation:

$$s_f = \sum_r \left(p_{(0,1)}(r) + p_{(1,1)}(r) \right) = \sum_r \left(1 - \frac{(1 + q_a) q_b}{q_a + q_b} \right)$$

A.2 Proof of Equation (3.6)

For $i \in \{1, \ldots, R\}$, let Y_i^k the random variable whose value is 1 if content i has been drawn from the catalogue after k steps and 0 otherwise. Y_i^k is a Bernoulli variable with parameter $1 - (1 - q(i))^k$, and by definition, $|A_1(k)| =$

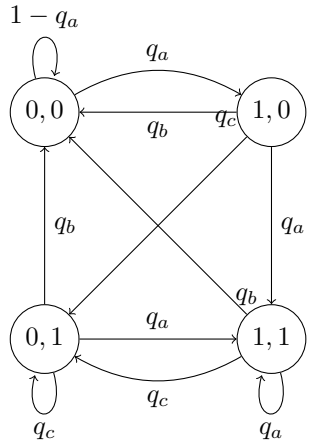


Figure A.1 – Discrete Time Markov Chain for the filter-cache interaction in the Fog (Fig. 4 of [169]). Each state correspond to content presence in the filter (first coordinate) and the Fog cache (second coordinate).

$\sum_{i=1}^{R} Y_i^k$ (as long as A_1 has not been reset). Let K_1 be the random variable giving the time necessary to fill the filter:

$$K_1 = \min\{k : |A_1(k)| \geq n_a\}$$

and let $\widehat{k_1} = f^{-1}(n_a)$. We want to show that K_1 is close to $\widehat{k_1}$, with high probability. More precisely, let $\epsilon > 0$, we want to find κ such that:

$$\mathbf{P}\left[|K_1 - \widehat{k_1}| \geq \kappa\right] \leq \frac{2}{R^\epsilon}$$

According to Chernoff bounds [229], we know that $\sum_{i=1}^{R} Y_i^k$ is close to $\mathbf{E}[\sum_{i=1}^{R} Y_i^k]$ with high probability. Precisely, for $\gamma \in (0, 1]$:

$$\mathbf{P}\left[|A_1(k)| \geq (1+\gamma)f(k)\right] \leq \exp(-\gamma^2 f(k)/3) \tag{A.1}$$

$$\mathbf{P}\left[|A_1(k)| \leq (1-\gamma)f(k)\right] \leq \exp(-\gamma^2 f(k)/2) \tag{A.2}$$

Let κ_1 be such that:

$$\left(\frac{f(\widehat{k_1})}{f(\widehat{k_1} - \kappa_1)} - 1\right)^2 \frac{f(\widehat{k_1} - \kappa_1)}{3} \geq \epsilon \log R \tag{A.3}$$

Then, with $\gamma = \frac{f(\widehat{k_1})}{f(\widehat{k_1} - \kappa_1)} - 1$ and $k = \widehat{k_1} - \kappa_1$ in Equation (A.1), we obtain:

$$\begin{aligned}
\mathbf{P}[K_1 \leq \widehat{k_1} - \kappa_1] &= \mathbf{P}[|A_1(\widehat{k_1} - \kappa_1)| \geq n_a] \\
&= \mathbf{P}|A_1(\widehat{k_1} - \kappa_1)| \geq (1+\gamma)f(\widehat{k_1} - \kappa_1)|] \\
&\leq \exp(-\gamma^2 f(\widehat{k_1} - \kappa_1)/3) \\
&\leq \exp(-\epsilon \log R) = \frac{1}{R^\epsilon}
\end{aligned} \tag{A.4}$$

Similarly, let us consider κ_2 such that:

$$\left(1 - \frac{f(\widehat{k_1})}{f(\widehat{k_1}+\kappa_2)}\right)^2 \frac{f(\widehat{k_1}+\kappa_2)}{2} \geq \epsilon \log R \tag{A.5}$$

We then take $\gamma = 1 - \frac{f(\widehat{k_1})}{f(\widehat{k_1}+\kappa_1)}$ and $k = \widehat{k_1} + \kappa_2$ in Equation (A.2) to obtain (recalling that $f(\widehat{k_1}) = n_a$):

$$\mathbf{P}[K_1 \geq \widehat{k_1} + \kappa_2] \leq \frac{1}{R^\epsilon} \tag{A.6}$$

Taking $\kappa = \max\{\kappa_1, \kappa_2\}$ and combining Equation (A.4) and Equation (A.6) gives, using the union bound $\mathbf{P}[A \cup B] \leq \mathbf{P}[A] + \mathbf{P}[B]$:

$$\mathbf{P}\left[|K_1 - \widehat{k_1}| \geq \kappa\right] \leq \frac{2}{R^\epsilon}$$

as desired.

For practical purposes, if $\kappa_1 \ll \widehat{k_1}$, it is possible to use a Taylor approximation of f:

$$f(\widehat{k_1} - \kappa_1) = f(\widehat{k_1}) - \kappa_1 f'(\widehat{k_1}) + \mathcal{O}(\kappa_1^2)$$

The left-hand-side of Equation (A.3) then becomes:

$$\frac{\kappa_1^2 f'(\widehat{k_1})^2}{3f(\widehat{k_1})} + \mathcal{O}(\kappa_1^3)$$

Thus in practice, Equation (A.3) can be fulfilled by taking:

$$\kappa_1 \approx \frac{\sqrt{3n_a\epsilon \log R}}{f'(\widehat{k_1})}$$

Similarly, Equation (A.5) can be fulfilled by taking:

$$\kappa_2 \approx \frac{\sqrt{2n_a\epsilon \log R}}{f'(\widehat{k_1})}$$

which justifies taking $\kappa = \max\{\kappa_1, \kappa_2\} \approx \frac{\sqrt{3n_a\epsilon \log R}}{f'(\widehat{k_1})}$.

A.3 Proof of Equation (3.7)

As $R \gg 1$, $q(i) \ll 1$ and thus $(1-q(i))^k = (1-(kq(i))/k)^k \approx \exp(-kq(i))$. Thus:

$$f(k) \approx \sum_{i=1}^{R} [1 - \exp(-kq(i))] = \sum_{i=1}^{R} \left[1 - \exp(-\frac{k}{iH_{R,\alpha}})\right]$$

and finally:

$$f(k) \approx \int_1^R \left[1 - \exp(-\frac{k}{xH_{R,\alpha}})\right] \mathrm{d}x$$

Computing the derivative $\widehat{f}'$ of this approximation yields:

$$\widehat{f}'(k) = \int_1^R \frac{1}{xH_{R,\alpha}} \exp(-\frac{k}{xH_{R,\alpha}})\mathrm{d}x$$

We will now distinguish between two cases, depending on whether $\alpha = 1$.

A.3.1 The case $\alpha = 1$

We have, in this case:

$$\widehat{f}'(k) = \frac{1}{H_{R,1}} \left[E_1(\frac{k}{RH_{R,1}}) - E_1(\frac{k}{H_{R,1}}) \right]$$

where $E_1(x) = \int_{-x}^{+\infty} \frac{e^{-t}}{t} dt$ is the exponential integral function. As $E_1(x) \approx 0$ for $x \gg 1$ and $E_1(x) \approx -\log x - \gamma$ for $x \ll 1$ (where $\gamma \approx 0.577$ is the Euler-Mascheroni constant), if we assume [1] $H_{R,1} \ll k \ll RH_{R,1}$, we obtain:

$$\begin{aligned} \widehat{f}'(k) &\approx \frac{1}{H_{R,1}} \left[-\gamma - \log(\frac{k}{RH_{R,1}}) \right] \\ &= \frac{\log(RH_{R,1}) - \gamma}{H_{R,1}} - \frac{\log k}{H_{R,1}} \end{aligned}$$

Using $H_{R,1} \approx \log R + \gamma$ yields:

$$\widehat{f}'(k) \approx \left(1 + \frac{\log\log R - 2\gamma}{\log R + \gamma} \right) - \frac{\log k}{\log R + \gamma}$$

Straightforward integration leads to:

$$f(k) \approx \underbrace{\left(1 + \frac{1 + \log\log R - 2\gamma}{\log R + \gamma} \right)}_{A} k - \underbrace{\frac{1}{\log R + \gamma}}_{B} k \log k$$

which is an approximation of the form $f(k) \approx Ak - Bk\log k$ with $A, B > 0$.

A.3.2 The case $\alpha \neq 1$

We have, in this case:

$$\widehat{f}'(k) = \frac{1}{k\alpha^2} \left[x\alpha \exp(-\frac{k}{x^\alpha H_{R,\alpha}}) - xE_{1+1/\alpha}(\frac{k}{x^\alpha H_{R,\alpha}}) \right]_{x=1}^{R}$$

where $E_n(x) = \int_1^{+\infty} \frac{e^{-xt}}{t^n} dt$ is the generalized exponential integral function. Again, we use $E_{1+1/\alpha}(x) \approx 0$ and $\exp(-x) \approx 0$ for $x \gg 1$. Thus:

$$\widehat{f}'(k) \approx \frac{R}{k\alpha} \left[\exp(-\frac{k}{R^\alpha H_{R,\alpha}}) - \frac{1}{\alpha} E_{1+1/\alpha}(\frac{k}{R^\alpha H_{R,\alpha}}) \right]$$

1. $H_{R,1} \ll k \ll RH_{R,1}$ is a reasonable assumption. $H_{R,1} \approx \log R + \gamma$ is usually smaller than 20. On the other hand, $RH_{R,1} \gg R \gg n_a$. As shown by eq. (3.7), k stays in the same order of magnitude as n_a and thus $k \ll RH_{R,1}$.

Furthermore, $E_{1+\beta}(x) \approx \frac{1}{\beta} + x^\beta \Gamma(-\beta) + \frac{1}{1-\beta} x$ for $x \ll 1$ and $0 < \beta < 2$, where Γ is the gamma function defined by $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} \mathrm{d}x$. Using $\beta = \alpha^{-1} \in [0, 2]$ (since we assume $\alpha > 1/2$) gives:

$$\begin{aligned}\widehat{f'}(k) &\approx \frac{R}{k\alpha} \left[1 - \frac{k}{R^\alpha H_{R,\alpha}} - 1 - \frac{1}{\alpha} \Gamma(-\frac{1}{\alpha}) \left(\frac{k}{R^\alpha H_{R,\alpha}} \right)^{1/\alpha} \right. \\ &\qquad \left. - \frac{1}{\alpha - 1} \frac{k}{R^\alpha H_{R,\alpha}} \right] \\ &= \frac{-\Gamma(-\frac{1}{\alpha})}{\alpha^2 H_{R,\alpha}^{1/\alpha}} k^{1/\alpha - 1} + \frac{R^{1-\alpha}}{(1-\alpha) H_{R,\alpha}}\end{aligned}$$

Integrating that equation yields:

$$f(k) \approx - \underbrace{\frac{\Gamma(-\frac{1}{\alpha})}{\alpha H_{R,\alpha}^{1/\alpha}}}_{B} k^{1/\alpha} + \underbrace{\frac{m^{1-\alpha}}{(1-\alpha) H_{R,\alpha}}}_{A} k \tag{A.7}$$

which is of the general form $f(k) \approx Ak - Bk^{1/\alpha}$. Equation (A.7) can then be further approximated by using the following approximation for $H_{R,\alpha}$:

$$H_{R,\alpha} \approx \begin{cases} \frac{R^{1-\alpha} - 1}{1-\alpha} & \forall \alpha < 1 \\ \zeta(\alpha) - \frac{1}{(\alpha-1) R^{\alpha-1}} & \forall \alpha > 1 \end{cases}$$

where $\zeta(s) = \sum_{i=1}^\infty i^{-s}$ is the Riemann zeta function.

A.4 Numerical evaluation of $t_C(r)$

As an example, Figure A.2 depicts $t_C(r)$ and $h(r)$ for a catalogue of $R = 10^7$ objects distributed according to a Zipf distribution with $\alpha = 1$. It can indeed be visualized that the regions where $t_C(r)$ varies slowly and $h(r)$ is close to one are mutually exclusive.

The validity of Che approximation is quantified in Figure A.3, which depicts the exact value $h(r)$ from Equation (3.9) versus its Che approximation from Equation (3.10) (where t_C is found by solving Equation (3.11)), for the same parameters as in Figure A.2. It can be seen that the relative error is smaller than 1%, and for objects that belong to the tail of the catalogue – thus having a little impact when computing the hit rate over the whole catalogue.

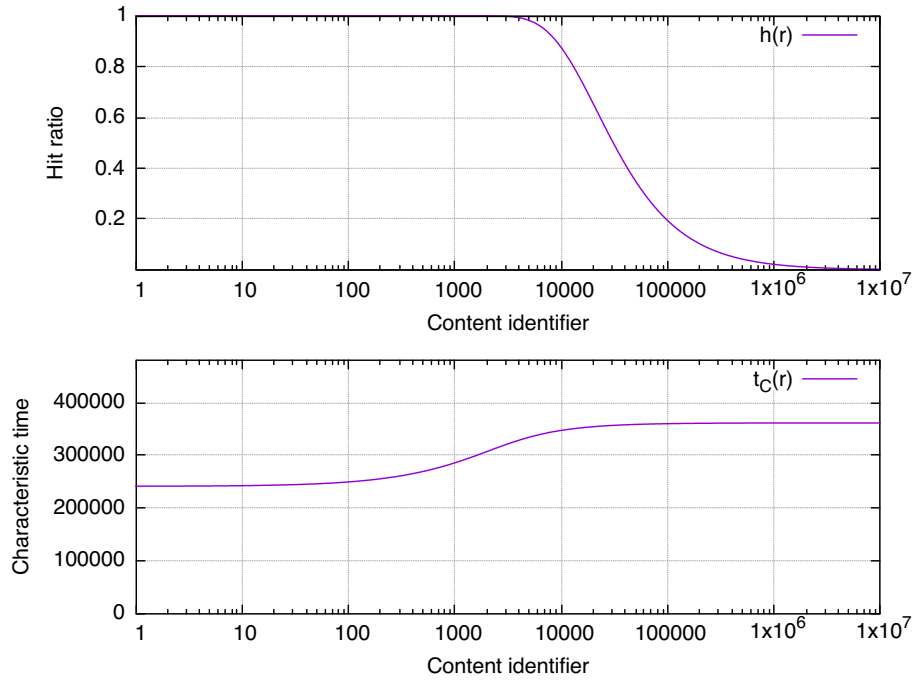


Figure A.2 – Che's approximation illustration: $h(r)$ and $t_C(r)$ for a Zipf catalogue with $\alpha = 1$ and $R = 10^7$. In the region where $t_C(r)$ evolves steeply ($r \leq 5000$), we have $h(r) \approx 1$ and thus the precise value of $t_C(r)$ has little impact.

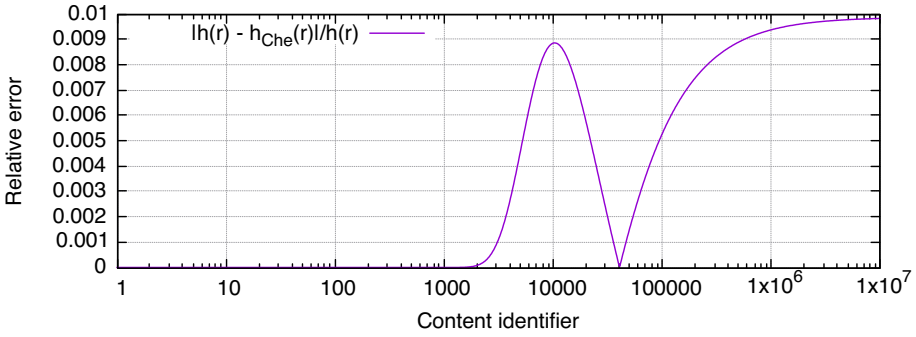


Figure A.3 – Che's approximation validation: exact value of $h(r)$ from Equation (3.9) versus Che approximation from Equation (3.10). Zipf catalogue with $\alpha = 1$ and $R = 10^7$.

Appendix B

Résumé étendu en Français

B.1 Introduction

Les dix dernières années ont vu l'émergence, sous le nom d'*"Internet des Objets"* (IoT) de multiples applications : villes intelligentes [11–13], voitures [9] et bâtiments [2–4] connectés, etc. Ces domaines, malgré leur multiplicité, sont des expressions de la même tendance qui consiste à munir des objets du quotidien d'une connexion internet. Les applications IoT présentent des défis considérables pour les réseaux de communications en termes de capacité, de flexibilité, ou de qualité de service. Pour répondre à ces défis, de nouvelles architectures réseau ont été mises en avant telles que les réseaux centrés contenu (ICN). Cette thèse s'inscrit dans cette lignée en enquêtant sur l'acheminement de paquets dans l'Internet des Objets centré contenu.

Les applications IoT sont fondés sur le même fondamental principe : d'automatiser ou d'optimiser une partie de notre environnement en y déployant des machines, capables non seulement de mesurer certaines variables environnementales (les *capteurs*) mais aussi d'agir dessus (les *actuateurs*). Ces machines sont dotées d'une connectivité sans-fil (par exemple le standard IEEE 802.15.4) et forment ainsi un réseau - l'Internet des Objets. Les applications IoT peuvent ensuite utiliser cette connectivité pour effectuer une boucle de contrôle qui prend en entrée les mesures des capteurs et sort des instructions pour les actuateurs. Ce fonctionnement général peut être utilisé pour représenter des boucles d'une complexité variable, du réglage d'un radiateur connecté à partir d'un thermomètre au contrôle des systèmes de direction d'un véhicule à partir de données produites par des centaines de capteurs embarqués.

Les particularités de l'Internet des Objets posent des défis particuliers aux réseaux, en particulier en termes de passage à l'échelle, de flexibilité mais aussi d'adaptation à des modes de communications différents [23]. Pour cette raison, des architectures et protocoles réseaux alternatifs ont été proposées pour l'IoT, et en particulier les réseaux centrés contenus (ICN) [40]. ICN est un paradigme réseau nouveau qui repose sur l'utilisation de noms porteurs de sens comme couche centrale du réseau. Ces noms sont des identifiants, qui décrivent l'objet (la donnée ou le service par exemple) accédé sur le réseau, se distinguant ainsi des traditionnelles adresses IP qui indiquent la location dudit objet. Ils sont utilisés avec un modèle de transmission par tirage : pour récupérer une donnée,

un client émet un paquet appelé "Intérêt" qui est acheminé jusqu'à un endroit où la donnée est stockée. Un paquet dit "Donnée" est utilisé pour transporter la donnée jusqu'au client en utilisant le chemin inverse de l'Intérêt correspondant. Le passage d'une approche centrée sur la localité à une approche centrée sur le contenu apporte de nombreux avantages théoriques, décrits dans [63] : l'unification de la gestion des couches applicatives et réseaux, la possibilité de découvrir des services dans le réseau, ou le support natif de la diffusion multipoint et du cache au niveau réseau, etc.

Néanmoins, si beaucoup d'études se sont penchées sur ces avantages et sur l'application d'ICN à des cas d'usage spécifique, il manque dans la littérature une étude systématique d'une question fondamentale : *comment gérer l'acheminement des paquets dans un réseau ICN utilisé pour l'IoT ?* En effet, les réseaux ICN-IoT posent des défis bien particuliers à l'acheminement de paquets en termes de passage à l'échelle, de flexibilité face à des réseaux très dynamiques ou des applications à forte contrainte de latence, ou de sélection du meilleur chemin quand un objet est disponible à plusieurs endroits du réseau. Les travaux de cette thèse se penchent sur cette question, en regardant l'acheminement de paquets à travers l'ensemble de l'architecture IoT. En particulier, les contributions présentées ici se répartissent en trois grandes parties. D'abord, la possibilité d'utiliser un acheminement géographique et sécurisé dans les réseaux de capteurs sans fil est évaluée. Ensuite, des stratégies d'acheminement sont proposées avec pour objectif de contrôler le temps de réponse d'une application déployée sur une plateforme de type Fog. Enfin, une architecture logicielle et réseau pour gérer conjointement réseaux et déploiement d'applications virtualisées à partir d'intentions abstraites est présentée.

B.2 Acheminement géographique dans les réseaux de capteurs sans-fil

Dans un premier temps, le problème de l'acheminement dans les réseaux de capteurs sans-fil est étudié. En effet, de précédents travaux ont montré des avantages d'ICN par rapport aux protocoles standards basés sur IPv6 [24] en termes d'utilisation mémoire et de consommation énergétique sur les capteurs, ce qui est crucial pour des capteurs à faible puissance. Néanmoins, ces avantages ne viennent pas sans de nombreux défis, et en particulier le problème d'acheminer les paquets efficacement avec le moins de trafic de contrôle possible.

Pour répondre à ces défis, cette thèse contient une étude de la possibilité d'utiliser de l'acheminement géographique [107]. L'acheminement géographique consiste à assigner à chaque paquet une destination géographique (par exemple, avec des coordonnées de géolocalisation) qui sera utilisé ensuite dans chaque nœud relais pour choisir le prochain saut. La plupart de ce type d'algorithme est fondée sur deux modes de relais : la sélection gloutonne (lequel de mes voisins est le plus proche de la destination du paquet) et une technique d'évitement des puits (situation dans laquelle le nœud relais est le plus proche de la destination du paquet dans son voisinage). Une revue des techniques d'évitement de puits est présentée dans [107].

B.2.1 L'architecture SLICT

Afin de mettre en œuvre l'acheminement géographique dans un réseau centré contenu de capteurs sans fil, l'architecture SLICT (pour objets localisées, sécurisées et centrés contenu) est introduite. Cette architecture repose sur quatre piliers : un protocole d'association sécurisée, un protocole de balisage sécurisé, un algorithme d'acheminement géographique, et une pile ICN implémentée dans le système d'exploitation RIOT.

Association sécurisée entre capteurs

Le but d'un protocole d'association sécurisé entre capteurs est d'assurer que l'ensemble des appareils qui participent au réseau sans-fil à plusieurs sauts sont des nœuds de confiance. Pour ce faire, une approche dite "*réseau de confiance*" est utilisée : chaque paire de capteurs en ligne de mire l'un de l'autre (i.e., physiquement capables de communiquer) effectue un protocole d'authentification mutuelle. Ainsi, Compagno et al., ont proposé *OnboardICNg*, un protocole d'association qui repose sur de la cryptographie symétrique. Dans cette thèse, un protocole similaire est proposé, fondée lui sur de la cryptographie asymétrique (plus adaptée à l'authentification mais plus coûteuse pour des capteurs à énergie limitée). La comparaison de ce protocole à OnboardICNg met en lumière un compromis intéressant. D'un côté, l'utilisation de la cryptographie asymétrique permet d'échanger moins de message et d'avoir un échange local aux deux nœuds s'authentifiant alors que la cryptographie symétrique nécessite de se connecter à une tierce partie qui sert d'ancre mutuelle de confiance. De l'autre côté, malgré l'apparition récente de modules physiques pour faire de la cryptographie asymétrique à moindre coût, OnboardICNg est beaucoup moins coûteux en termes d'énergie et de latence.

Balisage sécurisé

Le balisage permet aux nœuds d'informer régulièrement leurs voisins de leur position actuelle. C'est un mécanisme essentiel pour pouvoir réaliser l'acheminement géographique. Néammoins, les réseaux ICN sont fondés sur une approche par tirage alors que le balisage est une opération de poussée de données. Pour résoudre ce conflit, SLICT utilise des entrées persistantes dans la PIT, crée à la suite du protocole d'association sécurisée. Afin de garantir la sécurité de ce balisage, les balises sont encryptées avec une clé créée pendant le processus d'association.

Algorithme d'acheminement géographique

SLICT utilise le protocole GPSR [108] pour acheminer les paquets géographiquement. Ce protocole a deux principaux modes de fonctionnements : le routage glouton, où chaque nœud choisit son voisin le plus proche de la destination comme prochain relais d'un paquet, et le routage dit de *périmètre*, utilisé pour sortir des puits. Si le lecteur est renvoyé à [108] pour plus d'information sur GPSR, il est important de noter que le routage de périmètre nécessite de transporter de l'état à l'intérieur de chaque paquet, encodé sous la forme d'un champ Type-Longueur-Valeur (TLV).

Pile ICN dans le système d'exploitation RIOT

L'ensemble de ces modules a été codé dans une pile ICN dans le système d'exploitation RIOT [58]. En particulier, pour permettre la cohabitation entre diverses stratégies d'acheminement, le préfixe `/g/` est utilisé pour tout routage géographique. Ce préfixe est associé dans la FIB ICN à une entrée virtuelle qui représentent tous les voisins capables de faire du routage géographique. Le prochain saut est choisi parmi ces voisins en utilisant GPSR dans la couche de stratégie d'acheminement.

B.2.2 Évaluation de l'acheminement géographique

Pour évaluer une stratégie d'acheminement, deux critères principaux sont utilisés : la faisabilité (en termes de mémoire et de calcul sur les capteurs) et l'efficacité (en termes de coût énergétique de l'apprentissage des chemins). Ces coûts sont calculés à l'aide d'un modèle mathématique. En particulier, le modèle est construit pour deux stratégies principales : l'acheminement géographique (telle qu'introduit dans SLICT), et l'acheminement de type "*inonde-et-apprends*", correspondant à la référence dans la littérature sur ICN-IoT.

Inonde-et-apprends

L'approche dite "Inonde-et-apprends" (I&A) est l'approche la plus fréquemment trouvée dans la littérature ICN-IoT [24, 134–136]. Elle repose sur l'inondation du réseau pour des Intérêt ICN pour lesquels aucune route n'existe. La route est apprise grâce à la réponse au dit Intérêt : le voisin qui transmet le paquet Donnée correspondant (i.e., portant le même nom ICN) est enregistré comme prochain saut pour ce nom.

Pour compenser la naïveté de l'approche inonde-et-apprends, une version optimisée du processus d'inondation est aussi considérée. Elle utilise l'algorithme MPR (multi-point relais) [137] qui permet d'optimiser l'inondation en ne choisissant qu'un sous-ensemble des voisins pour propager les Intérêts pour lesquels aucune route n'est établie.

Modèle mathématique

Un modèle mathématique est proposé pour représenter la performance des trois stratégies considérées : GPSR, I&A, et MPR. Ce modèle dépend de trois variables principales : le taille du réseau (le nombre de noms ICN à supporter), la densité du réseau (le nombre moyen de voisins de chaque capteur), et son dynamisme (la fréquence avec laquelle une route pour un nom donné change). Il est construit en deux parties : la première modélise le coût de relai d'un Intérêt pour un nœud donné en fonction de la stratégie d'acheminement, et la deuxième regarde le coût global d'acheminement en étudiant comment les Intérêts se propagent dans le réseau sans-fil. En particulier, pour les stratégies I&Aet MPR, une campagne de simulation est menée pour mieux représenter le coût énergétique du processus d'inondation.

Résultats principaux

Le modèle permet de montrer deux résultats principaux. D'abord, en termes de faisabilité, l'acheminement géographique offre des avantages en termes d'utilisation mémoire pour les topologies denses et/ou larges. Ceci est dû à la localité de la stratégie d'acheminement, qui ne demande que de connaître des informations locales (i.e., la position des voisins) plutôt que des informations globales (un prochain saut pour chaque nom supporté sur le réseau IoT). De plus, l'acheminement géographique offre des avantages pour les réseaux très dynamique grâce à son trafic de contrôle local moins couteux que l'inondation – même optimisée avec MPR – du réseau de capteurs sans-fil. Pour les réseaux plus stables, l'état additionnel dans le paquet nécessaire pour GPSR rend l'acheminement géographique marginalement plus couteux en énergie que MPR.

B.3 Contrôle d'admission pour applications à temps de réponse contraint

Dans un second temps, le problème du contrôle d'admission dans des plateformes de type Fog est traité. En effet, pour être utiles, les données produites par les capteurs doivent pouvoir être traitées en temps limité. Dans ce but, le *Fog* a été inventé [17]. Il s'agit d'une plateforme de calcul et de stockage distribué se situant au bord du réseau, proche dans l'endroit où les données IoT sont produites et consommées afin de réduire la latence. Néanmoins, et en comparaison avec le Cloud, le Fog manque d'élasticité. Il est donc important de contrôler la charge du Fog pour garantir le temps de réponse de l'application. En particulier, cette section contient une proposition de contrôle d'admission de requêtes dans un nœud de type Fog, conçue pour optimiser l'utilisation des ressources sous contrainte de temps de réponse.

B.3.1 Contrôle d'admission dans le Fog et modélisation

La situation considérée est la suivante : un utilisateur envoie une requête pour une certaine donnée IoT. Pour produire cette donnée, il est nécessaire de récupérer une ou plusieurs mesures brutes de capteurs et d'effectuer des calculs sur ces mesures brutes. Comme souvent dans les interactions hommes-machine, l'utilisateur attend une réponse dans une latence de l'ordre de 100 ms [34]. Deux solutions de calcul et de stockage sont disponibles : le Fog, qui fournit une capacité de calcul et de cache fixe mais sans frais, et le Cloud, qui fournit une capacité de calcul et de cache infinie mais avec un coût qui croit en fonction de leur utilisation. Afin de répartir les requêtes entre Fog et Cloud, une stratégie de contrôle d'admission est déployée devant le Fog. Pour chaque requête, la stratégie décide ou bien d'accepter la requête dans le Fog, ou bien de la rediriger vers le Cloud. La problématique à résoudre est donc la suivante : *comment trouver une stratégie qui minimise les coûts de Cloud tout en gardant le temps de réponse moyen sous une limite donnée ?*

Pour répondre à cette problématique, un modèle mathématique est proposé, fondée sur un réseau de file d'attentes. Chaque file du modèle est choisie à partir d'exemples pertinents dans la littérature. En particulier, le réseau est représenté par une file M/G/1-PS, représentant le partage du réseau entre les différents

flux [149, 163, 164]. Pour les plateformes de calcul, le Fog est représenté par une file M/G/1-PS, qui représente sa capacité finie [149, 162], tandis que l'élasticité parfaite du Cloud est représentée par une file M/G/∞. Enfin, la stratégie de contrôle d'admission est représentée par une fonction $\phi : r \rightarrow [0, 1]$ qui à une requête pour une donnée r associe une probabilité d'être accepté dans le Fog. Le modèle de file d'attente peut ainsi être utilisé pour calculer le temps de réponse moyen et le coût associé à chaque stratégie.

B.3.2 La stratégie LRU-AC

Il y a trois chemins possibles pour une requête dans le système Fog-Cloud : le Cloud, le cache du Fog, ou le système de calcul du Fog. Comme le système de calcul du Fog a une capacité fixe, l'unique solution pour augmenter la capacité globale de Fog est d'augmenter le taux de succès du cache. Pour ce faire, une stratégie de contrôle d'admission pour le Fog est de sélectionner uniquement les requêtes pour des données populaires, qui auront une probabilité plus élevée d'être dans le cache.

Pour faire cette sélection, il faut donc être capable de (i) identifier la donnée cible pour chaque requête et (ii) d'en extraire des prédictions de popularités. La solution présentée dans cette thèse, appelée le LRU-AC, remplit ces deux conditions, avec l'avantage supplémentaire d'avoir une implémentation performante en termes de débit et de latence car elle peut être réalisée sur du matériel de type FPGA [153].

Identification des données cibles

Pour identifier les données cibles, le LRU-AC repose sur une architecture ICN spécifique, appelée *hICN* (pour Hybride-ICN) [154]. hICN est une architecture ICN construite dans IPv6/TCP, c'est à dire que les champs des en-têtes IP et TCP sont utilisé pour fournir une sémantique ICN. En particulier, les données sont nommées en utilisant (pour un Intérêt) l'adresse IPv6 de destination sous la forme d'une location (préfixe de 64 bits) et d'un identifiant (suffixe de 64 bits). Cette architecture est particulièrement intéressante pour le cas d'usage du contrôle d'admission en FPGA, car elle permet d'accéder à une sémantique applicative dans l'en-tête réseau grâce à des champs qui ont une position et une taille constante. En particulier, cela permet de réutiliser tous les outils développés pour IP comme la plateforme P4 [155].

Prédiction de popularité

Pour prédire la popularité des requêtes, le LRU-AC utilise un filtre LRU, i.e., un méta-cache sur les *identifiants* des données avec la politique d'éviction dite de l'objet utilisé le moins récemment (LRU). En effet, la présence (resp. absence) d'un identifiant dans un tel cache indique qu'il s'agit probablement d'une donnée populaire à accepter dans le Fog (resp. impopulaire à rediriger dans le Cloud). En particulier, le comportement d'un tel cache peut être prédit en fonction de la distribution de popularité des requêtes grâce à l'approximation de Che [165], ce qui permet d'intégrer cette stratégie dans le modèle de file d'attente présenté plus haut.

Pour implémenter le filtre LRU en FPGA, le LRU-AC utilise des filtres de Bloom vieillissant (ABF) [152]. Grâce à un modèle mathématique, il est montré qu'il est possible de paramétrer un ABF pour avoir un comportement identique à un filtre LRU. Une implémentation du LRU-AC est donc proposée en FPGA, ce qui permet d'obtenir un module de contrôle d'admission avec un débit de ligne de 16 Mpps et un temps de traitement d'une requête de moins de 3 µs.

B.3.3 Principaux résultats

Pour analyser les performances du LRU-AC, deux méthodes sont employées. Tout d'abord, le modèle mathématique est soumis à un processus d'optimisation pour comparer les performances moyennes du LRU-AC par rapport à une stratégie omnisciente et une stratégie naïve. Ensuite, pour comprendre le comportement plus précis du LRU-AC, une campagne de simulation est menée. En particulier, l'évaluation montre que le LRU-AC fonctionne quasiment aussi bien qu'une stratégie omnisciente (avec connaissance parfaite de la distribution de popularité). De plus, l'étude des distributions de temps de réponse simulé du système valide le choix d'implémenter le filtre LRU avec un ABF et montre un taux de dépassement de la limite de seulement 5%.

B.4 Orchestration d'applications et gestion de réseaux centrés contenus

Enfin, la problématique de l'orchestration et de la gestion de réseaux IoT-ICN est abordée. Ceci est particulièrement important dans l'Internet des Objets centrés contenu, entre autres à cause de l'échelle et de la diversité des réseaux IoT, du manque d'outils existant dans le monde d'ICN, ou du fort potentiel d'interaction entre application et réseaux ICN qui requiert une unification de la gestion des réseaux et de l'orchestration des applications.

B.4.1 Orchestration fondée sur l'intention

La diversification grandissante des utilisations et des technologies réseaux a rendu la gestion de réseau trop complexe pour l'être humain. Pour répondre à ce défi, de nouveaux paradigmes de gestion sont apparus, en particulier les réseaux définis par logiciel (SDN) [185] avec leurs orchestrateurs centralisés et riches en fonctionnalités. En particulier, des interfaces fondées sur l'*intention* ont été développées [190]. Dans une plateforme intentionnelle, le gestionnaire de réseau spécifie des objectifs abstraits au lieu de spécifier la procédure nécessaire pour atteindre ces objectifs. La plateforme est ensuite capable de configurer et de réguler le déploiement réseau de manière à atteindre ces objectifs.

B.4.2 La plateforme vICN

Dans cette optique, la plateforme vICN est introduite dans cette section. Il s'agit d'une suite logicielle d'orchestration et de gestion de réseau ICN à partir de l'intention.

En particulier, vICN utilise un modèle d'intention qui permet à l'utilisateur de spécifier son intention avec une granularité sur-mesure, de l'intention

abstraite à la configuration détaillée. Ce modèle d'intention est central au fonctionnement de vICN. Il est fondé sur un modèle relationnel objet [206]. Ce modèle a pour principal composant les *objets*, définis comme un ensemble d'attributs typés et de méthodes. Les attributs typés sont ou bien de type standard (chaîne de caractère, entier, etc.) ou des références à d'autres objets. Ces objets bénéficient ainsi de l'ensemble des avantages de la programmation orienté objet (héritage, abstraction, composition). De plus, le modèle est augmenté par la définition de *relations* entre les objets, par exemple de dépendance. La combinaison de l'héritage et des relations entre objets est l'élément principal qui permet de transformer une intention abstraite en plan de configuration fonctionnelle.

Pour garantir la mise en place de la configuration et l'état du système, chaque objet instancié dans le modèle est associé à un automate, qui garde en mémoire l'état dans lequel se trouve la ressource déployée. Les transitions entre états dudit automate correspondent à des actions sur le réseau – par exemple la création d'un conteneur Linux qui fait passer de l'état d'initialisation à l'état de création. Chacun des attributs de l'objet est aussi muni de son propre automate. L'utilisation d'automates permet de fournir des garanties en termes d'exécution d'un modèle. En particulier, les actions sur le réseau sont effectuées par de multiples fils d'exécution. La répartition des tâches entre fils d'exécution est faite par un ordonnanceur qui utilise le modèle pour apprendre l'ordre entre les différentes tâches à effectuer.

L'ensemble de ces mécanismes a été implémenté et est distribué sous licence libre dans le cadre du projet FD.io de la Linux Foundation [194]. Le code source, écrit en Python, contient une libraire fournie de ressources qui permettent de déployer des réseaux virtualisés complexes. En particulier, vICN a montré de bonnes performances sur un déploiement d'environ 50 nœuds virtualisés représentant un réseau de distribution de vidéo.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Routage centré-contenu pour l'Internet des Objets

Mots clés : Réseaux-Centrés contenus, Internet des objets, routage

Résumé : Les réseaux centrés contenus (ICN) sont considérés comme une solution aux nouveaux défis et modes de communication liés à l'émergence de l'Internet des Objets (IoT). Pour confirmer cette hypothèse, la problématique fondamentale du routage sur les réseaux ICN-IoT doit être abordée. Cette thèse traite de ce sujet à travers l'architecture IoT.
Premièrement, une méthode sécurisée est introduite pour acheminer des paquets ICN à partir de coordonnées géographiques dans un réseau sans-fil de capteurs à faible puissance. Elle est comparée à une inondation optimisée du réseau inspirée des approches existant dans la littérature. En particulier, leur faisabilité et passage à l'échelle sont évalués via un modèle mathématique. Le modèle est paramétré grâce à des données réalistes issues de simulation, de la littérature, et d'expériences sur des capteurs. Il est montré que le routage géographique permet de diviser la mémoire nécessaire sur les capteurs par deux et de réduire considérablement le coût énergétique du routage, en particulier pour des topologies dynamiques.
Ensuite, ICN est utilisé pour contrôler l'admission à une plate-forme de calcul de type Fog afin de garantir le temps de réponse. La stratégie de contrôle d'admission proposée, le LRU-AC, utilise l'algorithme Least-Recently-Used (LRU) pour apprendre en direct la distribution de popularité des requêtes. Son efficacité est démontrée grâce à un modèle fondé sur un réseau de files d'attente. Une implémentation du LRU-AC est proposé, utilisant des filtres de Bloom pour satisfaire aux contraintes des cartes FPGA. Son bien-fondé est prouvé par un modèle mathématique et son efficacité en termes de latence et débit démontrée.
Enfin, on présente vICN, un outil pour la gestion et la virtualisation de réseaux ICN-IoT. Il s'agit d'une plate-forme qui unifie la configuration et la gestion des réseaux et des applications en exploitant les progrès des techniques d'isolation et de virtualisation. vICN est flexible, passe à l'échelle, et peut remplir différents buts : expériences à grande échelle reproductibles pour la recherche, démonstrations mélangeant machines émulées et physiques, et déploiements réels des technologies ICN dans les réseaux IP existants.

Title : On Information-Centric Routing and Forwarding in the Internet of Things

Keywords : Information-Centric Networking, Internet of Things, Forwarding

Abstract : As the Internet of Things (IoT) has brought upon new communication patterns and challenges, Information-Centric Networking (ICN) has been touted as a potential solution. To confirm that hypothesis, the fundamental issue of routing and forwarding in the ICN-IoT must be addressed. This thesis investigates this topic across the IoT architecture.
First, a scheme to securely forward ICN interests packets based on geographic coordinates is proposed for low-power wireless sensor networks (WSN). Its efficiency is compared to an optimized flooding-based scheme similar to current ICN-WSN approaches in terms of deployability and scalability using an analytical model. Realistic data for the model is derived from a mixture of simulation, literature study, and experiments on state-of-the-art sensor boards. Geographic forwarding is shown to halve the memory footprint of the ICN stack on reference deployments and to yield significant energy savings, especially for dynamic topologies.
Second, ICN is used to enhance admission control (AC) to fixed-capacity Edge-computing platforms to guarantee request-completion time for latency-constrained applications. The LRU-AC, a request-aware AC strategy based on online learning of the request popularity distribution through a Least-Recently-Used (LRU) filter, is proposed. Using a queueing model, the LRU-AC is shown to decrease the number of requests that must be offloaded to the Cloud. An implementation of the LRU-AC on FPGA hardware is then proposed, using Ageing Bloom Filters (ABF) to provide a compact memory representation. The validity of using ABFs for the LRU-AC is proven through analytical modelling. The implementation provides high throughput and low latency.
Finally, the management and virtualization of ICN-IoT networks are considered. vICN (virtualized ICN), a unified intent-based framework for network configuration and management that uses recent progress in resource isolation and virtualization techniques is introduced. It offers a single, flexible and scalable platform to serve different purposes, ranging from reproducible large-scale research experimentation to demonstrations with emulated and/or physical devices and network resources and to real deployments of ICN in existing IP networks.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France