Computer Networks 270 (2025) 111494

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey paper

Exploring SDN architectures for IoT over Low-power and Lossy Networks (LLNs): A survey

Liticia Djennadi [a],*, Gladys Diaz [a], Khaled Boussetta [a], Christophe Cerin [b]

[a] Laboratoire de Traitement et Transport de l'Information, Institut Galilée, Université Sorbonne Paris Nord, 99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France

[b] Laboratoire d'Informatique de Paris Nord & DATAMOVE INRIA, Institut Galilée, Université Sorbonne Paris Nord, 99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France

ARTICLE INFO

Keywords:
IoT
SDN
LLNs
Architectures
Network management

ABSTRACT

While the Internet of Things (IoT) is transforming our daily lives, it still poses significant challenges in non-industrial use cases, such as smart homes or public facilities, where Low-power and Lossy Networks (LLNs) are commonly used. These technologies are cost-effective, but also raise important management issues. Software-Defined Networking (SDN) has emerged as a promising paradigm to address these challenges by introducing flexible and centralized network management. Although several SDN-based architectures have been proposed for IoT in non-industrial LLN environments, no comprehensive and focused survey has yet consolidated the recent advancements in this field. This paper discusses the need for SDN in IoT over LLNs, exploring how traditional approaches fall short and why SDN is essential for efficient network management. We also analyze existing SDN architectures for LLNs, propose an updated taxonomy, and identify research gaps and future challenges. This work serves as a valuable reference for researchers and practitioners aiming to adopt SDN on LLNs networks.

1. Introduction

Technological advances have significantly influenced the evolution of society, guiding it toward increasingly automated and intelligent processes. These innovations aim to simplify daily life while opening up new opportunities and offering unprecedented services. The Internet of Things (IoT) is one of the most striking examples of this transformation [1]. Since its inception, it has revolutionized several sectors and continues to unlock new possibilities.

In urban settings [2], the IoT enables efficient management of energy, water, and mobility, which is crucial due to rapid population growth. On a domestic level [3], the automation of everyday tasks through IoT brings tangible benefits in terms of comfort and energy efficiency. Thus, the IoT impacts all key aspects necessary for a modern society, providing solutions that improve quality of life, service efficiency, and resource management.

However, IoT ecosystems face several challenges, one of the most significant being their siloed structure. Each infrastructure is designed for a specific service, which lacks the flexibility to support multiple functions or applications. In a smart home environment, as shown in Fig. 1, safety, health, home automation, and telemetry applications typically function in isolation, with each system operating independently. This siloed structure limits the potential for cross-application interaction and data sharing. Unified analysis of data from multiple sources could significantly improve overall efficiency. For example, temperature sensors, which are commonly used for home automation tasks such as controlling smart thermostats, could also contribute to optimizing energy consumption in telemetry systems. By breaking down these silos and facilitating seamless integration, the full potential of IoT could be unlocked, transforming the home into a truly intelligent, interconnected environment. In addition, IoT raises even more challenges when it comes to Low-power and Lossy Networks (LLNs), as these networks introduce additional constraints [4]. Key challenges in LLNs include ensuring Quality of Service (QoS) in environments with unstable communication links, managing mobility as devices move within the network, maintaining robust security to protect against cyberattacks, and optimizing energy management for battery-powered devices. These issues are critical in enabling reliable and scalable IoT ecosystems, particularly in diverse and dynamic environments.

In response, Software-Defined Networking (SDN) has emerged as a key solution [5], as it has revolutionized networking by enabling centralized and intelligent management through the decoupling of control and data planes [6,7]. Although this idea dates back to the 2000s, SDN only gained popularity with the development of the OpenFlow

* Corresponding author.
E-mail address: liticia.djennadi@edu.univ-paris13.fr (L. Djennadi).

https://doi.org/10.1016/j.comnet.2025.111494
Received 14 March 2025; Received in revised form 2 June 2025; Accepted 20 June 2025
Available online 16 July 2025

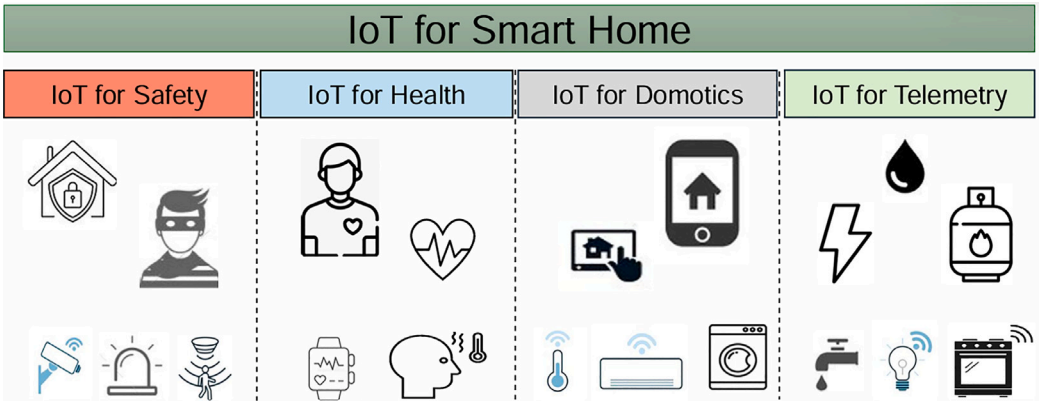


Fig. 1. Smart Home IoT applications.

protocol [8], which for the first time enabled standardized and direct communication between the two planes.

This concept was first adopted in the data centers of major tech companies and in optical networks, where its ability to optimize traffic management and improve network flexibility proved invaluable [9]. Building on these successes, SDN now appears to be a powerful solution to manage the complex and ever expanding ecosystem of connected devices in today's IoT-driven world. However, leveraging SDN in IoT environments introduces several challenges [10]. To address these issues, various SDN architectures have been specifically developed to address IoT systems' different requirements and constraints. As IoT continues to grow and evolve, these SDN-based solutions will play a crucial role in enabling the seamless integration and efficient operation of IoT systems across various domains.

Since each architecture targets specific challenges and has distinct characteristics, it is essential to conduct a comprehensive analysis of existing SDN solutions for IoT to fully leverage its potential. This analysis will provide a detailed overview of the various SDN architectures proposed for IoT networks, highlighting the progress made in tackling key challenges. By examining these architectures, we can better understand how each approach addresses unique IoT issues. Moreover, such an analysis is crucial to identify the gaps in current solutions and to suggest potential future research avenues. This overview will help identify areas that require further development and refinement.

While Industrial IoT (IIoT) is a specific subset of IoT with stricter requirements in terms of real-time operation, security, and fault tolerance [11], it has already been the subject of dedicated SDN surveys, such as a comprehensive review published in 2024 [12]. However, the architectural and operational differences between IIoT and general purpose IoT are substantial. SDN architectures designed for IIoT typically address highly controlled, mission critical industrial environments and are therefore not directly applicable to the more heterogeneous, dynamic, and consumer-oriented IoT systems.

Despite some efforts, SDN integration in non-industrial IoT, remains underexplored. For instance, [13] presents the applications of SDN and Network Virtualization (NV) in the context of IoT, providing a comprehensive overview of various implementation aspects within IoT systems, and briefly reviewing some existing SDN-based architectures. In [14], the authors discuss the challenges faced by IoT and the technologies capable of addressing them, including SDN. They also review several solutions for integrating SDN into IoT systems.

The authors in [15] introduce the concept of "Software-Defined Internet of Things" (SD-IoT), exploring SDN techniques tailored for IoT and proposing architecture-specific solutions. Similarly, [16] examines SDN and Network Function Virtualization (NFV) as core IoT technologies, addressing network complexity and adoption challenges.

Focusing on edge-cloud systems, [17] highlights the benefits of SDN/NFV integration, identifying application domains and challenges at the IoT edge. Meanwhile, [18] provides a taxonomy of SDN-based IoT frameworks that address issues such as fault tolerance, scalability, and security, while also identifying research gaps.

The authors in [19] review SDN security solutions in multiple IoT domains, while [20] categorizes SDN-based security strategies for fog computing environments in IoT. Finally, [21] explores SDN applications alongside emerging technologies such as blockchain and distributed computing, emphasizing future research directions.

Table 1 summarizes all these surveys on SDN for IoT over LLNs.

Although these works contribute valuable insights, they exhibit notable limitations. Some surveys focus on early technologies and initial SDN-IoT integration efforts [13,14], and thus fail to capture the progress and new architectural models developed in recent years. Since 2020, however, a substantial number of new architectures have been proposed, many of which are not covered in these surveys. Others emphasize specific architectural components or combinations, such as Network Virtualization or SDN/NFV integration [15–17], without offering a comprehensive analysis of SDN-based architectures themselves. Finally, many surveys focus on isolated challenges or application domains [18–21], lacking a unified and systematic framework to classify and compare existing SDN-IoT architectures.

Consequently, there is little critical analysis regarding the suitability of different architectures for various IoT contexts, leaving practitioners without clear guidance on selecting or adapting SDN solutions to modern IoT requirements.

Given the increasing interest in the convergence of SDN and IoT, particularly in the context of LLNs, our survey critically explores whether SDN offers unique advantages for IoT or if alternative solutions could achieve similar results. Furthermore, our survey addresses the previously mentioned gaps by providing an inclusive review of the SDN architectures proposed from 2015 to 2024. We adopt a broad approach, examining all architectures without applying restrictive selection criteria, and systematically classify, compare, and assess them. Taking this approach, our goal is to provide a thorough, up-to-date, and unbiased resource that reflects the latest advances and provides a clear analysis of the actual contributions of SDN to IoT innovation in LLNs.

We summarize the main contributions of this work as follows:

- In this survey, we explore and motivate the use of SDN in the IoT over LLNs, highlighting its necessity and benefits.
- We provide an analysis of existing SDN architectures for IoT, with a focus on LLNs, based on a review of the literature from 2015 to 2024, and we propose a new taxonomy.
- We identify open challenges and their potential solutions that need further investigation for the efficient use of SDN in IoT.

Table 1
Comparative study of related surveys for SDN-IoT.

Year	Reference	Description	Years covered
2016	[13]	Discusses the opportunities offered by the combination of SDN and NV within IoT, and the existing architectures	Not specified
2018	[14]	Presents the challenges faced by IoT and the technologies that can address them, including SDN and its integration solutions	Not specified
2018	[15]	Focuses on IoT challenges and how the use of SDN can address them	Not specified
2020	[16]	Discusses the challenges of using SDN for IoT networks	Until 2019
2021	[17]	Investigates the benefits and application domains of SDN and NFV	Not specified
2022	[18]	Categorizes SDN-based IoT frameworks targeting specific issues	2010–2022
2022	[19]	Focuses on SDN solutions targeting security in IoT networks	January 2016 - July 2021
2023	[20]	Evaluates SDN mechanisms for IoT-Fog protection	Not specified
2024	[21]	Focuses on SDN applications in IoT across specific network technologies	2014–2022
2025	Our survey	Provides an overview of existing SDN architectures for IoT over LLNs, highlighting their limitations and opportunities for future research	2015–2024

The remainder of this paper is structured as follows. Section 2 presents an overview of the concepts of IoT and SDN, providing the necessary background and motivation for their integration, with a focus on LLNs. Section 3 offers a comprehensive analysis of existing architectures, while Section 4 introduces a detailed taxonomy. Section 5 outlines key limitations and future challenges, and finally, Section 6 concludes the paper with a summary.

2. Background and motivation

This section provides the necessary background on IoT and SDN, along with the motivation to integrate SDN into IoT environments, highlighting its relevance and potential benefits for LLNs.

2.1. Internet of Things

The concept of the Internet of Things (IoT) was first introduced in 1999 as a network of connected objects, each uniquely identifiable through RFID technology [1]. Since then, this concept has evolved to encompass a wide range of interconnected devices capable of transmitting, receiving, and interacting with each other using embedded sensors, software, and other technologies [22].

A classical multi-tier IoT architecture [23], represented in Fig. 2, is composed of three layers:

Cloud Layer: Represents centralized data centers for large-scale data storage and processing, but introduces latency due to its distance from end-users.

Fog Layer: An intermediary layer between the cloud and IoT devices, bringing data processing and storage closer to users to reduce latency and support real-time applications.

Device Layer: It consists of physical objects that can capture various quantities, such as humidity, temperature, rotation, or gas concentration. This information can then be processed, stored, and wirelessly transmitted to a collector node (sink) or a gateway. To enable the transmission of the data captured by the sensor node to the sink or the gateway, various technologies can be short- or long-range [24].

- Short-range technologies: Short-range wireless technologies refer to communication systems that enable data transmission over short distances, typically ranging from a few centimeters to about 100 m. They are used in applications such as Bluetooth [25], Wi-Fi [26], and Zigbee [27], to connect nearby devices, such as IoT sensors, phones, and smart home devices.
- Long-range technologies: Long-range wireless technologies refer to communication systems that support data transmission over extended distances, often from hundreds of meters to several kilometers. They are commonly used in applications such as cellular networks (4G/5G) [28], and LoRaWAN [29], which allows connectivity for devices spread over large areas, such as in smart cities, agricultural monitoring, and remote IoT deployments.

In short-range wireless technologies, effective routing is essential to ensure reliable data transmission across connected devices networks. This is particularly significant in scenarios with limited coverage and constrained resources, namely Low-power and Lossy Networks (LLNs).

2.2. Low-power and Lossy Networks

Low-power and Lossy Networks (LLNs) are increasingly adopted in IoT applications due to their ability to connect a large number of low-power nodes, which is essential for large-scale IoT deployments. However, due to their limited radio coverage, multi-hop routing is commonly employed, where data are forwarded through intermediate nodes to reach its destination. However, given their characteristics, limited energy, unstable links, and dynamic network topologies, the use of specialized and efficient routing protocols is required.

DYMO-LOW [30], LOAD [31], and HYDRO [32] are examples of early routing protocols developed prior to the specification of LLNs routing requirements. These protocols were essentially simplified adaptations of existing designs and were unable to address the unique demands of LLNs. The Routing Over Low power and Lossy Networks (RoLL) working group later identified critical requirements for LLNs [33–35], which these protocols failed to meet [36].

To address the limitations of previous protocols, the RoLL working group initiated the development of RPL (Routing Protocol for Low-power and Lossy Networks) [37]. Introduced by the Internet Engineering Task Force (IETF) in 2012 with RFC 6550, RPL was designed as a standardized protocol for energy- and resource-constrained networks, such as sensor networks and IoT applications. Over time, RPL has become the de facto standard for IoT routing due to its adaptability,

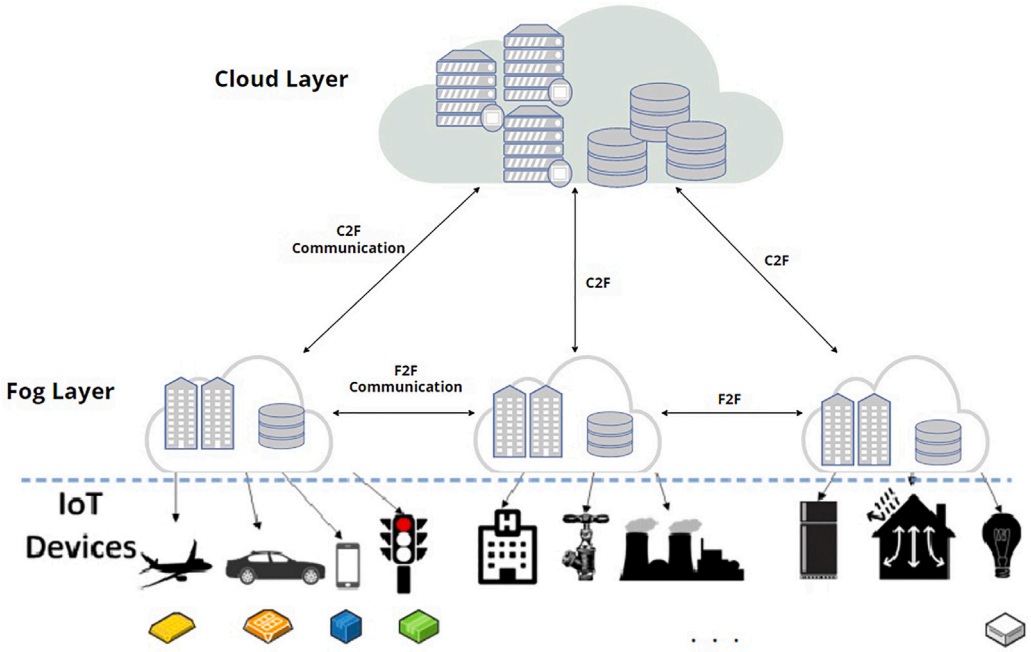


Fig. 2. Multi-tier IoT architecture.

extensive compatibility, and ability to tackle the unique challenges of LLNs.

In contrast, another protocol, named Thread [38], was proposed in 2014 by the Thread Group. It was designed to establish efficient routes for reliable data delivery while addressing energy efficiency and link stability in LLNs. The thread focuses on smart homes and building automation. It has been widely adopted in residential environments and consumer products, with support from major tech companies like Apple and Google.

While Thread is well-suited for residential use, its scalability and flexibility are limited, making RPL a more versatile option for broader IoT applications. Unlike Thread, which manages networks with only a few hundred nodes, RPL scales seamlessly to thousands of nodes. Furthermore, Thread's reliance on certified equipment for interoperability increases both complexity and cost. In contrast, RPL offers a streamlined design with simpler equipment requirements and enhanced scalability, making it the preferred routing protocol for diverse IoT deployments.

RPL organizes nodes into a Destination-Oriented Directed Acyclic Graph (DODAG) and builds routes based on an Objective Function (OF), such as OF0 or MRHOF, which rely on hop count and Expected Transmission Count (ETX), respectively. It operates using several control messages: *DODAG Information Object* (DIO) to advertise routing information, *Destination Advertisement Object* (DAO) to propagate upward routing information to parent nodes, *DAO Acknowledgement* (DAO-ACK) to confirm DAO reception, and *DODAG Information Solicitation* (DIS) to request DIOs from neighbors when joining the network.

To reduce the overhead caused by frequent DIO transmissions, RPL employs the Trickle algorithm, which dynamically adjusts the transmission rate based on the stability of the network [39]. Trickle divides time into intervals I, bounded by a minimum value $I_{\min}$ and a maximum value $I_{\max}$. At the start of each interval, a node listens to messages from its neighbors and maintains a counter c of consistent messages heard. A random transmission time T is selected within the second half of the interval, i.e., $T \in \left[\frac{I}{2}, I\right]$. If fewer than K consistent messages are heard ($c < K$), the node transmits a DIO message at time T; otherwise, it suppresses transmission, avoiding redundancy.

If the network remains stable over time, the interval length I is doubled (up to $I_{\max}$), thereby reducing the transmission frequency. Conversely, when inconsistency is detected, the interval resets to $I_{\min}$ to quickly propagate updated information.

This mechanism allows RPL to maintain low communication overhead during stable periods while ensuring rapid convergence under dynamic conditions, making it well-suited for LLNs and widely adopted in real-world IoT applications. Despite its strengths, RPL still faces significant challenges that impact the performance of IoT networks [40].

Choosing the appropriate Objective Function (OF) is critical as it can significantly impact network performance. Relying on a single metric to select the parent node may lead to overloading some nodes, causing congestion and poor load balancing. This, in turn, degrades QoS and shortens the network's lifespan.

In addition, while the Trickle Timer reduces control messages, it is not always optimal, especially under high-dynamic networks, where it can lead to increased energy consumption or poor performance. Moreover, the configuration of its parameters significantly affects energy consumption and performance [41,42], but these settings are often left at default values and are not adapted [37].

RPL is also vulnerable to security threats, such as malicious nodes that broadcast false DIO messages. These attacks disrupt trickle-timer functionality, destabilize the network, drain energy, and hinder route convergence, degrading both QoS and overall performance [43].

2.3. RPL enhancement

Given RPL's widespread adoption in IoT networks, several studies have been conducted to optimize this routing protocol and improve its efficiency. These efforts can be classified as follows:

- **Objective Function enhancement:**

Several objective functions that combine multiple metrics are available in the literature. In [44], the combined metrics objective function (COM-OF) integrates expected lifetime ELT, child count CC, expected transmission count ETX, and residual energy E_{res} to optimize energy consumption and prolong the useful life of the network. ELT predicts energy depletion time, CC prevents node overload by tracking child nodes, ETX ensures the reliability of the path and E_{res} enhances longevity. Similarly, ADAPTIVE-OF [45] focuses on balancing energy consumption and providing QoS by combining ETX and energy

Table 2
Overview of objective functions based on metrics combination.

OF	Aim	ETX	ELT	Q	RSSI	CC	N_b	E_{res}	HC	Improvement
COM-OF [44]	Load balancing	✓	✓	✗	✗	✓	✗	✓	✗	Reduces power consumption, improves PDR, extends network lifetime
ADDITIVE-OF [45]	Optimize energy consumption and provide QoS	✓	✗	✗	✗	✗	✗	✓	✗	Enhances PDR, optimizes energy consumption
rpl-TotEg-Neighbors [46]	Support mobility while maintaining performance	✓	✗	✗	✗	✗	✓	✓	✗	Improves PDR, reduces convergence time, lowers energy consumption
EEQ [47]	Enhance energy efficiency	✓	✗	✓	✗	✗	✗	✓	✗	Reduces network overhead
EL-RPL [48]	Extend network lifetime	✓	✗	✗	✗	✓	✓	✗	✓	Lowers energy consumption, balances load, enhances PDR
RM-RPL [49]	Improve mobility support	✗	✗	✗	✓	✗	✗	✗	✗	Improves packet delivery ratio
CQAOF [50]	Prevent congestion and ensure QoS	✓	✗	✓	✗	✓	✗	✓	✓	Improves congestion control, reduces packet loss

metrics, with weights assigned to each, enhancing performance in resource-constrained IoT networks.

Rpl-TotEg-Neighbors [46] considers the total energy, neighbor count N_b, and ETX. The total energy affects the longevity of the nodes, N_b shapes the network topology, and ETX guarantees reliable transmissions, thus improving load balance and network stability. In another approach, EEQ [47] calculates the rank of nodes based on ETX, energy consumption, and the length of the active queue Q. Q reflects the load of the node, allowing dynamic routing to avoid overloading the nodes with high utilization or low energy, ensuring more balanced network performance.

EL-RPL [48] also targets load balancing and energy optimization by combining ETX, load, and battery depletion index BDI. BDI tracks energy consumption, while the load metric manages the traffic distribution by considering the number of child nodes under each parent. EL-RPL reconfigures the DODAG to ensure balanced traffic and extend the life of the network. RM-RPL [49] improves parental selection in mobile networks by incorporating the received signal strength indicator $RSSI$ and proximity. $RSSI$ helps identify reliable parents, reducing packet loss and improving network stability in dynamic environments.

Finally, the CQAOF (Congestion and QoS-Aware Objective Function), as proposed in [50], is designed to optimize routing in IoT networks by addressing both congestion and QoS. It integrates key metrics, including residual energy, buffer occupancy, hop count HC, and ETX. This mechanism minimizes congestion while promoting efficient load distribution across the network.

Table 2 provides an overview of all these Objective functions.

- **Trickle algorithm enhancement:**

To improve the efficiency of the Trickle algorithm, several enhanced versions have been proposed. The Flexible Trickle algorithm (FL-Trickle) [51] optimizes energy consumption by increasing $I_{\min}$, fixing transmission time T to $\frac{I}{2}$, and allowing certain doubled intervals to be ignored or exceeded, thus making more efficient use of network resources. In a similar vein, the FI-Trickle algorithm [52] enhances load balancing and energy efficiency by doubling the DIO interval I only when a message is sent and resetting the counter c at T, which reduces redundant transmissions and improves fairness.

On the other hand, D-Trickle [53] adapts dynamically by adjusting the redundancy constant k based on the number of received messages and neighbors, ensuring better message propagation in less active areas while reducing unnecessary transmissions in busier regions. Elastic Trickle [54] further contributes to load balance by adjusting the listening period at the beginning of each interval I according to the density of the neighboring node, extending it in denser areas to better distribute messages across the network. Similarly, LA-Trickle [55] uses a learning automaton to adjust the number of repetitions of the minimum interval before doubling it, speeding up the resolution of inconsistencies through rewards and penalties.

Lastly, the Energy-Efficient Trickle algorithm (EE-Trickle) [41] reduces energy consumption by fixing T to $\frac{I}{4}$ for the first three transmissions, then selecting T randomly from $[\frac{I}{4}, I]$ for subsequent transmissions, ensuring long-term network stability and energy efficiency.

Table 3 provides an overview of all these variants of the Trickle algorithm.

- **Protocol enhancement:**

Many studies have focused on improving the RPL protocol as a whole, particularly its mechanism. The enhanced RPL protocol in [56] ensures stable connectivity by enabling parent nodes to monitor RSSI and alert mobile nodes of potential disconnections, prompting timely parental selection and optimizing energy usage. Similarly, RM-RPL [49] conserves energy by flagging mobile nodes that remain stationary for an extended period and halting unnecessary DIO broadcasts, while ensuring reliable delivery of critical packets through acknowledgments, retransmissions, or parental reselection.

CQARPL [50] addresses congestion through a bilateral decision-making mechanism that enables parent nodes to refuse new child nodes when saturated and dynamically adjusts routing based on the real-time buffer and congestion indicators. A multi-topology routing algorithm for RPL introduced in [57] differentiates IoT traffic according to QoS, assigning each traffic type (non-critical, critical, regular) to a virtual RPL instance with a tailored objective function and DODAG topology, optimizing resource management.

Table 3
Overview of Trickle algorithm variants.

Algorithm	Aim	Modified parameter(s)	Method	Improvement
FL-Trickle [51]	Reduce energy consumption and convergence time	Minimum sending Interval I_{min}, Transmission time T	Sets $I_{\min}$ higher, fixes T at $\frac{I}{2}$, allows skipping/exceeding some intervals	Reduces unnecessary transmissions
FI-Trickle [52]	Enhance energy usage and load balancing	Sending Interval I, counter c	Doubles I only if DIO sent, resets c at T instead of I expiration	Balances energy use across nodes
D-Trickle [53]	Improve load balancing	Redundancy constant K	Adjusts K based on neighbor messages, increasing in low-traffic areas and decreasing in busy ones	Avoids redundant messages in high-traffic areas
Elastic Trickle [54]	Dynamic load balancing	Listen-only period	Extends listen-only period as neighbor density increases	Distributes transmissions evenly in dense areas
LA-Trickle [55]	Speed up convergence	Minimum sending Interval I_{min}	Uses a learning automaton to adjust I_{min} repetitions based on network state	Reduces convergence time in dynamic conditions
EE-Trickle [41]	Extend network lifetime and reduce energy usage	Transmission time T	After convergence: fixes T at $\frac{I}{4}$ for the first three transmissions, randomizes T in range $[\frac{I}{4}, I]$ after that	Reduces computational overhead

Table 4
Overview of RPL proposed variants.

Variant	Aim	Method	Improvement
Hoghooghi et al. [56]	Ensure stable and efficient transmissions during node movements	Periodically measure RSSI to detect weak signal links, triggering a neighbor discovery process when a potential disconnection is detected.	Improves PDR, network overhead, end-to-end delay, and energy consumption.
RM-RPL [49]	Ensure reliable data delivery in dynamic environments	Uses CRITICAL-ACK messages for high-priority packets, with retransmissions or parent reselection if acknowledgments are missed.	Reduces handover delays, and optimizes energy consumption with low control overhead.
CQARPL [50]	Preventing congestion and ensuring QoS	Adds congestion management: parent nodes refuse new children when buffers are saturated, and DIO messages include congestion status.	Improves congestion control, QoS, reduces delay and overhead.
Bhandari et al. [57]	Providing QoS differentiation to meet diverse IoT application requirements	Segments the network into virtual RPL instances, each tailored to a specific QoS objective, with nodes selecting parents based on traffic type.	Improved QoS: reliability, delay, packet delivery, stability, with minimal overhead.
CoSec-RPL [58]	Detect non-spoofed copycat attacks to enhance security	Uses an IDS to flag attackers based on abnormal DIO message patterns.	Efficient detection and mitigation of the attacks with minimal overhead.
Mayzaud et al. [59]	Detect version number inconsistencies to identify potential attacks	Monitor version inconsistencies in RPL messages and flags nodes based on report analysis.	Effective detection of version number inconsistencies attacks.

The CoSec-RPL protocol [58] defends against "non-spoofed copycat" attacks, where a malicious node sends false copies of DIO messages, by using an Intrusion Detection System (IDS) based on statistical techniques that identify abnormal behavior and flag malicious nodes without overloading the network. Lastly, [59] presents a detection algorithm that identifies malicious behaviors by monitoring version inconsistencies in RPL messages. Nodes with suspicious inconsistencies are flagged, and a final check removes nodes considered suspicious by only a single monitoring node, improving detection accuracy.

Table 4 provides an overview of all these RPL variants.

Discussion:

Despite the benefits that the proposed objective functions provide, combining multiple metrics for path selection presents several limitations. Since each metric operates on a different scale, a normalization process is required to ensure compatibility and effective balance. Additionally, assigning weights to these metrics can be challenging, as the importance of one metric over another may vary depending on specific network conditions or requirements. This fixed weighting approach can lead to suboptimal decisions in varying scenarios, as it does not dynamically adapt to changes in network density, mobility, or traffic patterns.

For example, EEQ [47] calculates the rank of nodes based on ETX, energy consumption, and active queue length Q, ensuring a more balanced network performance. However, in scenarios with unstable link qualities or those that cannot tolerate packet losses, ETX becomes a critical metric, as poor link selection can significantly degrade overall performance. In such cases, assigning a greater weight to ETX ensures reliable communication. In contrast, in highly stable conditions, prioritizing energy consumption may be more beneficial to maximize the life

of the network. Relying on a fixed and rigid metric weighting does not account for these varying requirements, leading to potentially suboptimal path selections. To address this, more dynamic approaches that adapt the importance of metrics based on real-time network conditions are necessary to enhance RPL's adaptability and overall performance in diverse IoT environments.

Additionally, although Trickle variants improve the performance of RPL and outperform the basic Trickle algorithm, they still have limitations. Each variant is designed to address specific needs or traffic types, making it well-suited for certain use cases but not universally adaptable. For instance, FL-Trickle [51] optimizes energy consumption by increasing $I_{\min}$, fixing the transmission time T to $\frac{I}{2}$, and allowing certain doubled intervals to be ignored or exceeded, thus making more efficient use of network resources. However, this approach assumes a relatively stable network with predictable traffic patterns. In highly dynamic IoT environments, such as those with sudden traffic surges or frequent topology changes, the strategy of ignoring or exceeding certain intervals may lead to inconsistencies in control message propagation. These inconsistencies can result in delayed network convergence or uneven energy consumption across nodes, counteracting the intended energy-saving benefits and negatively impacting the networks performances. Thus, the specialization of the proposed variants can lead to performance degradation in scenarios outside the target conditions. Consequently, a dynamic approach that could adapt across diverse use cases and traffic types is needed to further enhance RPL's overall efficiency and resilience.

Similarly to the Trickle variants and combined metrics, different RPL variants are tailored to specific objectives and may not be suitable for all use cases. Applying them outside their intended contexts can result in significant performance degradation. For instance, some assumptions made by these variants may not hold universally. In [56], a decrease in RSSI is interpreted as an indicator of mobility, but this assumption may not always be accurate. RSSI fluctuations can also stem from interference that degrades link quality rather than actual movement. Furthermore, protocols incorporating anomaly detection, such as CoSec-RPL [58], have specific requirements regarding the number and optimal placement of monitoring nodes. An incorrect choice of node quantity or placement can compromise the protocol's effectiveness, leading to inefficiencies and potential blind spots in attack detection. While these variants bring valuable improvements in certain scenarios, they can pose substantial challenges when applied to more general or diverse IoT environments.

Thus, the proposed solutions remain rigid and lack the adaptability required to address the dynamic nature of IoT networks. This rigidity extends to the IoT infrastructure itself, which is often designed to support only a single service or application. Such inflexible architectures fail to accommodate the diverse and evolving requirements of modern IoT deployments.

Moreover, this lack of adaptability is further compounded by challenges in platform management, which has emerged as a significant limitation. Once deployed, managing IoT devices becomes a major challenge, primarily due to the need for manual intervention on each device. This process is both labor-intensive and time-consuming, particularly in large-scale deployments involving thousands of interconnected devices.

A critical issue in this context is the difficulty of performing remote software updates. The complexity of managing a large number of devices makes it impractical to perform individual updates manually, significantly hindering the scalability and efficiency of IoT networks, and emphasizing the need for automated management.

Therefore, significant challenges remain in IoT and, more specifically, in LLNs. These ongoing issues reveal critical gaps that require further exploration and innovative solutions to optimize network performance, resilience, and long-term sustainability. They highlight the need for greater adaptability, improved management, and autonomy to address efficiently the dynamic and diverse requirements of modern IoT deployments. A promising approach to tackle these challenges is the use of Software-Defined Networks (SDN).

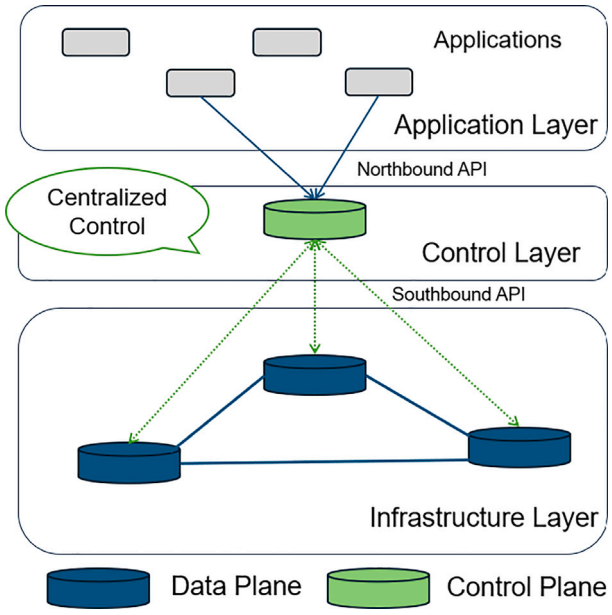


Fig. 3. SDN network architecture.

2.4. Software Defined Networking

Software-Defined Networking (SDN) is a new paradigm that allows centralized network management through a software entity called a controller. This controller provides a global network view, enabling real-time monitoring and dynamic configuration of network devices. Unlike traditional network architectures, where control is distributed across individual devices, SDN centralizes decision-making, optimizes performance, improves security, and facilitates the implementation of policy changes across the entire network. By decoupling the control plane from the data plane, SDN also simplifies the deployment of new services, making networks more adaptable to the evolving needs of applications.

Fig. 3 illustrates the architecture of an SDN network, structured into three distinct layers, with Application Programming Interfaces (APIs) facilitating communication between them. The roles and functions of each layer, as well as the APIs, are described below.

Infrastructure Layer: It consists of the physical and logical elements that provide the foundation of the SDN network. It defines the data plane functionality, which is made up of network devices such as switches and routers. These devices are responsible for forwarding and processing data packets within the network.

Control Layer: Responsible for managing and controlling the network, it centralizes and virtualizes the control functions, giving the SDN controller the ability to manage the entire network from a single point, making the network more agile and responsive to changes.

Several SDN controllers are presented in the literature, with a detailed analysis of 34 different controller architectures provided in [60]. However, some of these architectures were excluded from the in-depth analysis due to limited documentation or lack of updates, making them less relevant for modern network management practices. The study identifies several critical aspects that differentiate the controllers, including their modularity, architectural model (centralized or distributed), scalability, latency, and fault tolerance.

Among these characteristics, modularity stands out as a crucial feature for modern SDN controllers. This flexibility allows network functionalities to be customized, adapting the controller to meet specific needs. With modular controllers, modules can be easily added or removed based on system requirements, thereby simplifying both system extension and modification. Moreover, by offloading logic to

Table 5
Characteristics of SDN controllers.

Controller	Architectural model	Scalability	Fault tolerance	Latency	Modularity
OpenDaylight [61]	Centralized/Distributed	High	High	Low to Medium	High
Floodlight [62]	Centralized	Medium	Low (Single point of failure)	Low to Medium	Medium
ONOS [63]	Distributed	High	High	Low	High
Ryu [64]	Centralized/Distributed	Medium	Medium	Low to Medium	High
Trema [65]	Centralized	Medium	Low (Single point of failure)	Medium	High

dedicated modules, functions can be executed more rapidly, resulting in a more efficient overall architecture.

Concerning architectural models, in a centralized architecture, a single controller manages the entire network, offering a unified global view that simplifies network management. This makes it particularly well-suited for smaller networks where simplicity is key. However, as the network grows, scalability becomes a major concern. A centralized controller can become a bottleneck, unable to handle an increasing number of devices or high traffic volumes. Furthermore, it becomes a single point of failure, meaning that its failure can bring down the entire network, significantly impacting the fault tolerance of the system.

On the other hand, distributed architectures use multiple controllers working collaboratively, thus enhancing both scalability and fault tolerance. By distributing the workload across several controllers, these systems can handle large networks more efficiently. However, this distributed approach introduces challenges in maintaining state consistency and synchronization between controllers. This synchronization is essential to ensure that all controllers have an up-to-date and unified view of the network, but it can also introduce additional latency due to the need for constant communication and coordination among the controllers.

Ultimately, the choice between centralized and distributed architectures depends on the specific needs of the network. For small networks with simpler requirements, a centralized approach might be ideal, while larger, more complex networks tend to benefit from a distributed architecture. Table 5 summarizes the key characteristics of some of the most notable controllers from the study.

Application Layer: It refers to the applications and services that leverage the network to provide specific functionalities. It is where network policies, security rules, and Quality of Service (QoS) settings are defined. These applications interact with the control layer to request network resources or enforce network behavior.

A key feature of this layer is the use of dashboards and management interfaces, which provide administrators with a user-friendly way to monitor and control the network. These dashboards enable visualization of network performance, detection of anomalies, and application of configurations in real-time. Through intuitive graphical interfaces, operators can define policies, review logs, and analyze network metrics without directly accessing low-level configurations.

Northbound API: This API facilitates the exchange of information between the SDN controller and the applications running on top of the network. It enables developers to write custom applications and services that can take advantage of the control and programmability features offered by SDN, making the network more adaptable to application needs. Examples of Northbound APIs include RESTful APIs [66], widely used due to their simplicity and compatibility with web applications.

Southbound API: It plays a crucial role as the communication protocol between the SDN controller and the data plane (network devices). It allows the controller to manage data forwarding operations, receive network event notifications, collect statistical reports, and announce network capabilities to the underlying devices. One of the most commonly used Southbound APIs in SDN networks is OpenFlow [8], which provides a standardized way for controllers to communicate with switches and routers.

Thus, this paradigm offers key advantages such as flexibility, centralization, and programmability, all of which are essential for modern networks. These attributes are particularly crucial for addressing the unique challenges posed by IoT systems, including their constant evolution in terms of size and complexity. Moreover, SDN is already well-established in various domains, including Wide Area Networks (WANs) [67] and 5G [68], making it no longer just an emerging paradigm.

In WANs, SDN helps eliminate the need for physical relocation and manual interventions by enabling centralized control and dynamic network reconfiguration. Administrators can remotely perform network adjustments, reducing both the cost and time required for network management. In 5G, SDN plays a pivotal role in network slicing, allowing the creation of virtual networks tailored to specific applications and optimizing resource allocation to meet the demands of critical services.

Given its extensive benefits and proven success across multiple domains, exploring the application of SDN in IoT is imperative. However, this raises essential questions about its sustainability, the tangible benefits it can offer, and how it will address the key challenges of IoT networks.

2.5. Motivation for integrating SDN in IoT over LLNs

With its distinctive characteristics, SDN emerges as a highly suitable solution to address the unresolved challenges of IoT networks.

Currently, the Internet of Things still struggles to gain widespread adoption, largely because applications are deployed in isolated silos. Often, infrastructures are built specifically for a single use case or service, limiting their versatility. This rigidity complicates management, limits scalability, and hinders service evolution.

To overcome these limitations, sharing infrastructures and enabling different applications to leverage the same data is essential. In a smart home context, for example, and as shown in Fig. 4, breaking down silos allows temperature data (from thermal sensors) to be used simultaneously by multiple services: healthcare (monitoring patient health conditions), telemetry (energy optimization) and home automation (automatic heating and air conditioning regulation). Similarly, the presence sensor data can be utilized for both safety (intrusion detection) and telemetry (adjusting lighting and ventilation according to actual occupancy).

Through its centralization, SDN will facilitate large-scale data analysis and support the simultaneous operation of multiple applications. It will also enable the seamless integration of new services without the need to modify the existing network infrastructure. This approach enhances operational efficiency by providing an overview of resources, allowing for more effective management and informed decision-making through a unified perspective of the entire infrastructure.

Moreover, SDN provides a network abstraction layer, allowing the partitioning of the physical network into independent virtual networks, each with its own management and routing policies. This flexible approach encourages resource sharing and enables multiple services to coexist while meeting their specific requirements within a unified infrastructure. By optimizing resource utilization, SDN allows networks to adapt to evolving demands without the need for physical modifications. This approach maximizes the potential of IoT, ensuring its effective utilization in such environments.

In addition, SDN has the potential to improve platform management [69]. One of its key strengths lies in precise fault detection. Through continuous monitoring and real-time data collection from

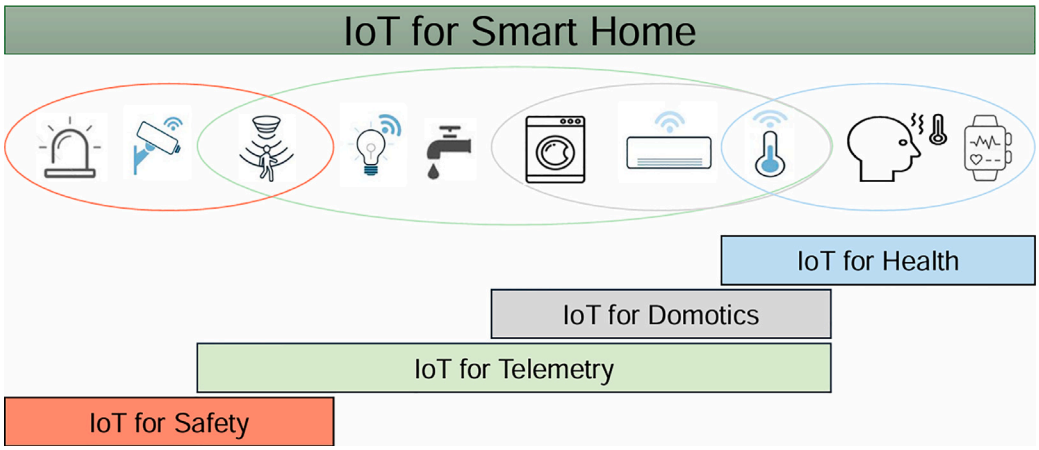


Fig. 4. Unifying Smart Home IoT applications.

network nodes, the SDN controller can promptly identify anomalies indicative of hardware failures, connectivity issues, or performance degradation. This proactive fault detection minimizes downtime by enabling rapid response and corrective actions, such as rerouting traffic or isolating faulty nodes.

It also facilitates seamless firmware updates across the network without requiring physical access to nodes. It allows for the remote activation or deactivation of nodes, dynamic reconfiguration of their roles, and even flashing or behavior modification to adapt to changing application needs. By combining real-time fault detection with centralized control, SDN enhances the reliability, efficiency, and adaptability of IoT platforms, ensuring optimized network performance under various conditions.

Furthermore, regarding the challenges posed by routing in LLNs, the contributions of SDN can be classified as follows:

- **Quality of Service:**

Providing quality of service (QoS) in LLNs remains a significant challenge, particularly with the RPL protocol. Originally designed for constrained environments, RPL prioritizes energy efficiency and routing simplicity but often falls short in addressing the evolving QoS requirements of modern IoT applications. These applications often require high reliability, low latency, or substantial bandwidth. However, these demands are not static; they vary over time based on specific application needs and the dynamic nature of the network. This variability makes it particularly challenging for RPL, in its standard form, to consistently meet such diverse and changing QoS requirements.

Effective QoS management requires adaptive mechanisms capable of allocating resources and prioritizing traffic in real-time to accommodate the ever-changing demands of applications. To achieve this, innovative enhancements are necessary to equip RPL with dynamic capabilities, enabling it to proactively adjust to the application's requirements. Such enhancements would involve modifying parameters like routing priorities, control messages frequency, and objective functions based on real-time network conditions and application-specific QoS needs. This highlights the need for centralized management, which can be provided effectively by SDN [70].

However, enabling SDN-based QoS control in LLNs also requires reliable and deterministic communication at the MAC layer. In this context, the IETF-standardized 6TiSCH architecture, built on top of the TSCH protocol, plays a key role as an enabler of SDN in LLNs [71]. By providing time-synchronized and collision-free communication, 6TiSCH establishes a stable network environment necessary for SDN to perform centralized scheduling and efficient resource control.

Therefore, integrating SDN with RPL becomes not only feasible but also significantly more effective when combined with 6TiSCH. This synergy enables real-time, fine-grained control over network operations, allowing for dynamic QoS optimization and better support for the diverse and evolving needs of modern IoT applications.

- **Mobility:**

Mobility presents significant challenges for IoT networks, requiring both accurate detection and efficient management to maintain connectivity and performance.

Mobility detection typically relies on monitoring variations in signal quality but relying solely on these indicators can lead to inaccuracies, particularly in LLNs. To improve detection, it is crucial to incorporate other methods, such as detecting a node sequentially by different nodes, indicating that the node is moving across the network. This sequence of detections suggests mobility, as the node's presence is observed in different locations over time. It is essential to analyze these sequential detections more thoroughly, considering factors such as the timing and consistency of the detections. This method can provide more reliable identification of mobile nodes compared to relying solely on signal quality variations, which may not always be accurate. However, this process requires centralized analysis, which is challenging due to the distributed nature of the network making it difficult to analyze sufficient data. SDN, through its centralized and up-to-date view of the topology, combined with real-time analysis of data from all nodes [72], can enable a more accurate mobility detection.

Once mobility is detected, efficient management becomes crucial to maintain stable connections as nodes move. Rapid reselection of the parent node demands intensive signaling, which requires reducing the sending intervals of signaling messages. Although this improves the convergence time, it significantly increases signaling overhead and, consequently, energy consumption. To achieve a better balance between energy efficiency and performance, SDN, with its centralized control, plays a vital role by enabling selective signaling adjustments. For instance, signaling efforts can be concentrated only on nodes likely to serve as the parent of the mobile node, thereby reducing unnecessary signaling overhead and energy usage.

This adaptive approach ensures optimal performance and energy conservation, making SDN an essential enabler of effective mobility management in IoT networks.

- **Security:**

IoT networks are inherently vulnerable, as they can be targeted by multiple types of attacks. To ensure optimal network performance, it is crucial to implement mechanisms capable of detecting anomalies, identifying attackers, and deploying countermeasures to mitigate these threats. While numerous solutions have been proposed in the literature, offering certain advantages, their effectiveness remains limited [73].

This limitation arises from the reliance on distributed analysis, which complicates the accurate identification of attackers and increases the likelihood of false positives, ultimately affecting the stability of the network.

In LLNs, this issue is further exacerbated by the limited memory and processing capabilities of the devices, which complicates the implementation of robust cryptographic mechanisms and other security protocols. SDN can play a pivotal role in addressing these challenges by leveraging its centralized control and global network view. By analyzing traffic patterns and node behavior in real-time, SDN enables more accurate anomaly detection and attacker identification [74], reducing false positives and ensuring a more effective response to threats.

- **Energy management:**

Energy management is crucial for extending the lifetime of LLNs, with two primary challenges: achieving effective load balancing and minimizing energy consumption.

Achieving load balancing is essential, as uneven load distribution can lead to network fragmentation or even complete disconnection. Traditional load-balancing mechanisms, which often depend on distributed decision-making, lack the holistic perspective needed to optimize resource allocation effectively. In contrast, SDN enables dynamic adaptation to variations in load and connectivity, enhancing network longevity [75]. By considering multiple factors, such as residual energy, the number of neighbors, and child nodes, SDN leverages its centralized, real-time view of the network to implement a more effective and adaptive load-balancing strategy. This comprehensive approach ensures an equitable distribution of load, optimizing resource usage and maintaining a sustainable balance across the network.

To reduce energy consumption, it is essential to optimize both data and signaling transmissions. For data messages, techniques such as data aggregation [76] can help decrease the number of transmissions by avoiding redundancy. The selection of routing paths is designed so that nodes transmitting similar data share the same relaying aggregator node, allowing only an optimized version of the data to be transmitted. SDN, with its centralized control and global network view, enables dynamic path selection to ensure that data flows from nodes with semantically similar information are directed toward appropriate aggregation points [77]. This approach reduces redundancy and minimizes the energy spent on transmitting duplicate data, thereby extending the network's lifetime.

For signaling, adapting it to traffic characteristics is crucial for achieving optimal energy consumption. SDN, with its centralized control and real-time awareness of network state and traffic patterns, can dynamically adjust the sending intervals of signaling messages, for example, during low-traffic periods or stable network conditions, allowing the network to operate in an energy-saving mode without compromising performance.

However, although SDN can reduce overall signaling, the control traffic it generates can affect network performance. This is because the SDN controller continuously collects network information to maintain a comprehensive view, introducing overhead that is particularly problematic in low-power IoT environments.

For this reason, and given the strong need to leverage SDN for IoT, particularly in LLNs, several SDN architectures have been specifically designed to account for the characteristics of Low-power and Lossy Networks. The following section provides a comprehensive analysis of existing architectures for LLNs, highlighting their distinctive features, innovations, and advantages.

3. SDN architectures for IoT

The integration of SDN into IoT networks has given rise to various architectures tailored to the specific needs of these networks. Fig. 5 presents a chronological overview of major SDN-based IoT architectures, highlighting their year of introduction and their evolutionary relationships. These architectures differ in focus, addressing key LLNs challenges such as routing optimization, energy efficiency, and security. A brief description of each follows, outlining its core innovations and contributions.

SDN-WISE and its evolved versions

SDN-WISE [78], proposed in 2015, is one of the first true implementations that extends the OpenFlow protocol for IoT networks. This stateful architecture allows IoT nodes to maintain local flow tables, which store flow rules that define how incoming and outgoing packets should be processed. Each flow rule consists of match conditions (such as packet headers), actions to be performed (such as forwarding or dropping the packet), and counters for tracking packet statistics (like the number of packets processed).

The flow table in SDN-WISE is designed to reduce the dependency on the central controller by allowing the nodes to process packets independently for a given period. Once a flow rule is installed in a node's flow table, it remains valid for a predefined duration, which is configurable depending on the network requirements. This local storage of flow rules minimizes the number of requests sent to the controller, thereby reducing overhead and enhancing network scalability. When a new flow is encountered that does not match an existing flow rule, the node will send a request to the controller to install a new flow rule.

SDN-WISE operates with an external controller, which provides centralized network management and topology awareness. The interaction between nodes and the controller is facilitated through the OpenFlow protocol [8], ensuring seamless communication across diverse devices. For routing control messages between nodes, SDN-WISE uses a custom protocol, while Dijkstra's algorithm is employed by the controller to determine the optimal routing paths for data.

SD-WISE [79] is an improved version of SDN-WISE, designed to overcome its limitations by integrating the ONOS controller and extending its capabilities. Like SDN-WISE, it is a stateful architecture where nodes maintain internal state to guide network behavior. However, SD-WISE introduces a more dynamic and context-aware state management, allowing nodes to adjust their state in response to environmental or network changes for more adaptive control.

A key enhancement is the energy optimization mechanisms included in SD-WISE, essential for battery-constrained Wireless Sensor Networks (WSNs). It employs duty cycle management (i.e., controlling node sleep and wake cycles) and transmission power control, adjusting node behavior based on conditions such as link quality or traffic load, features that were limited in SDN-WISE.

In terms of security, SD-WISE integrates Trusted Platform Modules (TPM) to enforce context-based rules through trusted hardware. This ensures that critical operations, like software updates or executing actions, are only allowed if validated by the hardware, strengthening node integrity and policy enforcement. With these features, SD-WISE offers a more secure, adaptive, and energy-efficient architecture than its predecessor.

In 2023, a new architecture named STEER [80] was proposed to enhance SDN capabilities in IoT by integrating SDN-WISE with Intention-Driven Networks (IDN). Its main innovation is the addition of an IDN-for-IoT layer between the control and application layers, enabling applications to express high-level intents that are automatically mapped to appropriate network behaviors.

Unlike traditional approaches that rely on manual configuration through graphical interfaces, STEER automates network configuration by interpreting and executing actions based solely on the defined intent of the application. Its selection algorithm operates in two phases: exploration, where each behavior is tested and evaluated, and exploitation, where the most effective behavior is applied.

This phased approach enables STEER to adapt dynamically and optimize resource utilization, offering a significant improvement over

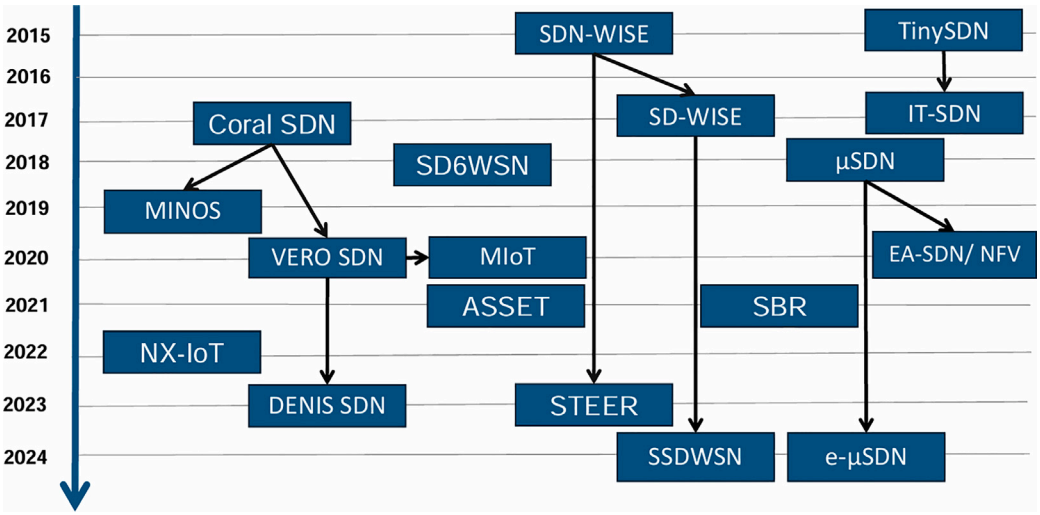


Fig. 5. Chronological overview of SDN architectures for IoT.

static configurations in IoT networks while better aligning with the high-level intents of applications.

More recently, SSDWSN [81], an architecture derived from SD-WISE, was proposed to tackle key SDN challenges in IoT, such as control overhead, congestion, and network lifetime. It preserves SD-WISE's core features while integrating a Deep Reinforcement Learning (DRL) agent to predict network state and optimize decision-making for improved performance.

Specifically, the authors implemented two algorithms based on Proximal Policy Optimization (PPO): PPO-NSFP (Network State Forecasting Policy) and PPO-ATCP (Adaptive Traffic Control Policy).

PPO-NSFP aims to reduce the control overhead associated with frequent network state updates. It operates by alternating between observation and prediction phases. During the prediction phase, the controller temporarily suspends the collection of network state information. Once the phase ends, this collection resumes, and the controller compares the predicted state with the actual one. This feedback is used to reward or penalize the model, enabling it to adjust its parameters and improve the accuracy of future network state predictions.

On the other hand, PPO-ATCP reduces congestion and extends network lifetime by minimizing transmissions through data aggregation and load balancing. After observing the network, the controller makes decisions and sends actions to the nodes. Then, the collection of state information is paused during a prediction phase, where the impact of these actions is forecasted. Once resumed, the predicted and actual states are compared to update the model and improve its decision-making capabilities in future cycles.

Thus, the integration of DRL in SSDWSN demonstrates a promising approach to proactively manage the network and enhance its overall performance.

TinySDN and its evolved version

TinySDN [82] is an SDN architecture introduced in 2015 that proposes an innovative approach by supporting the coexistence of multiple SDN controllers within the same network. It relies on the Collection Tree Protocol (CTP), originally developed for the TinyOS operating system [83], along with a custom communication protocol between nodes and controllers. To simplify its implementation, and given the relatively simple experimental scenarios evaluated, the control logic was executed directly on the sensor nodes. This approach was feasible due to the minimal computational load required.

An improved version of the TinySDN architecture, called IT-SDN (Improved TinySDN) [84], was later introduced with the goal of providing a more modular architecture, independent of the operating system. One of its key improvements is the clear separation between the different protocols used, which enhances modularity and maintainability. A notable feature of IT-SDN is its support for source routing, not only for data packets but also for control messages. This allows the controller to configure flow rules across multiple nodes using a single packet, significantly reducing control message overhead.

In both TinySDN and IT-SDN, flow rules are defined by the controller and stored locally on the nodes in flow tables, enabling centralized control while maintaining responsive and programmable network behavior. Importantly, neither architecture includes predefined management policies. Instead, they focus solely on enabling communication between nodes and controllers, leaving developers the flexibility to implement their own management strategies according to the specific needs of their applications.

Coral SDN and its evolved versions

Tryfon Theodorou et al. [85] introduced Coral SDN, an SDN-based architecture explicitly designed to enhance RPL's performance in dynamic IoT environments. This architecture focuses on addressing the limitations of RPL in mobility scenarios (e.g. smart cities) and point-to-point (P2P) communications, which are critical for environments such as underground mines or industrial setups.

As shown in Fig. 6, Coral SDN builds on RPL for routing integrates a custom external controller and a specific API to dynamically configure key parameters of RPL. At its core, the architecture introduces two topology discovery algorithms:

TC-NR (Topology Control - Node Request): An active algorithm where the controller requests nodes to gather information from their neighbors to construct a detailed network map.

TC-NA (Topology Control - Node Announce): A passive approach in which nodes report their neighbor information periodically during routine announcements, reducing overhead in stable networks.

The architecture enables the adaptive configuration of Trickle timer parameters, a process referred to as **Moderate RPL**, specifically adjusting the minimum interval I_{min} based on node mobility levels. For highly mobile nodes, I_{min} is reduced to allow more frequent control message exchanges, improving route stability and ensuring seamless mobility management.

In addition, Coral SDN proposes a new objective function for RPL to enhance P2P communications. This function prioritizes paths that explicitly include the destination node, optimizing latency and reliability for direct communications. The process of changing the objective function is referred to as **Deep RPL**.

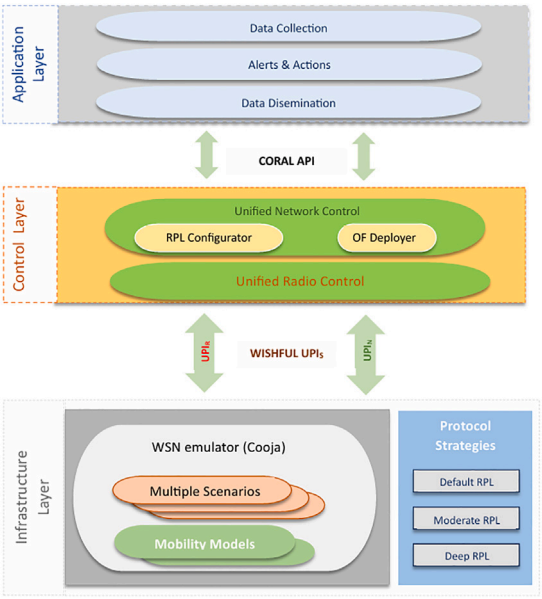


Fig. 6. Coral SDN architecture.

To further support dynamic IoT environments, Coral SDN offers centralized management through a graphical user interface (GUI). This GUI allows users to configure routing metrics (e.g., link quality) and select between routing methods such as next-hop only (ideal for scenarios with limited node mobility, such as a single moving node) and full path (to minimize flow table requests from intermediate nodes) enabling fine-tuned control of routing behavior while minimizing resource usage on constrained devices.

By combining these capabilities, Coral SDN significantly enhances RPL's adaptability and efficiency, demonstrating robust performance in networks characterized by high mobility or challenging communication requirements.

MINOS [86] is a multi-protocol SDN architecture for IoT, proposed in 2019, as an enhanced version of Coral SDN. This architecture addresses the limitations of relying on a single protocol for diverse IoT services by enabling a dynamic selection of routing protocols based on application-specific requirements. MINOS integrates two main protocols: Dynamic RPL and Coral SDN. Dynamic RPL extends the traditional RPL protocol by allowing the dynamic configuration of various parameters, such as the objective function, to improve adaptability. Coral SDN complements this by offering two control algorithms and supporting routing strategies like full-path or next-hop routing, which are tailored to different network scenarios.

In MINOS, the selection and configuration of the most appropriate protocol are guided by the user via the GUI integrated with the SDN controller. This approach ensures flexibility in adapting the network to the specific requirements of different IoT services. By supporting multiple protocols, MINOS offers a versatile and efficient solution to address the diverse demands of IoT networks.

VERO SDN [87], another enhanced version of Coral SDN, was specifically designed to address the overhead issues that Coral SDN may introduce in resource-constrained environments. One of the main innovations in VERO SDN is the use of a second, long-range radio interface that is dedicated exclusively to the transmission of control messages. This approach bypasses the need for a routing protocol for control message transmission, as the long-range radio ensures that control messages can be sent directly and efficiently without relying on a routing protocol.

For data message routing, VERO SDN continues to utilize the topology discovery and control mechanisms established in Coral SDN, maintaining compatibility with the existing network structure. This ensures that VERO SDN can still leverage the full flexibility of Coral SDN's routing capabilities while significantly reducing the overhead associated with routing control messages.

By using a separate interface for controlling traffic, VERO SDN enables better network performance and reduced congestion, making it particularly useful in scenarios where minimizing overhead is crucial, such as in large-scale IoT deployments.

MIoT [88], introduced in 2020, builds on VERO SDN by introducing advanced mechanisms tailored to the demands of mobile environments. This architecture focuses on addressing mobility-related challenges, while simultaneously optimizing resource usage and minimizing communication overhead. The key innovation in MIoT lies in its intelligent mobility detection algorithm, which enables automatic detection of node mobility without requiring user intervention. This is particularly beneficial in dynamic environments where devices, such as sensors, may frequently change locations, ensuring seamless connectivity and efficient routing.

MIoT inherits the core characteristics of VERO SDN, including the use of a secondary long-range radio interface for the transmission of control messages, which helps reduce overhead. However, MIoT introduces an intelligent mobility detection mechanism that is central to its operation. The architecture detects the movement of nodes in real time, adapting the network configuration accordingly to ensure optimal performance even in the face of constant node mobility. This makes MIoT particularly suitable for environments such as extreme sports or amusement parks, where IoT devices are highly mobile, and the network must continuously adjust to ensure reliable communication.

The dynamic nature of MIoT's design ensures that it remains efficient in highly mobile and resource-constrained scenarios while maintaining low overhead. This makes MIoT a promising solution for applications where mobility is a key factor, such as real-time monitoring in large, changing environments.

DENIS SDN [89] is another SDN architecture designed to address the challenges of high-density IoT networks, specifically those with node densities ranging from 100 to 1000 per square kilometer. It inherits the core features of VERO SDN but introduces several enhancements to ensure efficient communication and minimize packet loss in these densely populated networks.

The primary approach of DENIS SDN involves the use of network slicing, which can be either logical or physical. Logical network slicing is implemented by selecting different routing paths for each slice, depending on the type of traffic being transmitted. This allows the network to handle different traffic types more efficiently, ensuring that high-priority or time-sensitive traffic is given precedence over less critical data. On the other hand, physical network slicing assigns a separate radio channel to each network slice to avoid interference between slices. This physical separation helps maintain the integrity and reliability of communications in crowded network environments. This architecture provides a robust solution for high-density IoT applications, ensuring stable and efficient performance even in challenging network conditions.

ASSET

ASSET [90] is an SDN architecture for IoT, proposed in 2021, aimed at mitigating specific vulnerabilities of the RPL protocol. These include attacks designed to exploit inconsistencies that rapidly deplete nodes' batteries or disrupt the formation of the RPL tree through traffic spoofing, where malicious nodes deceive others into selecting them as their parent. This architecture, illustrated in Fig. 7, enables fault detection, identifies attackers, and implements appropriate mitigation strategies to protect the network.

The architecture operates in three adaptive monitoring modes. In Slim Mode, the network is stable, and the controller collects only basic topology data, while nodes monitor minimal metrics such as global repair counts, reporting anomalies when predefined thresholds

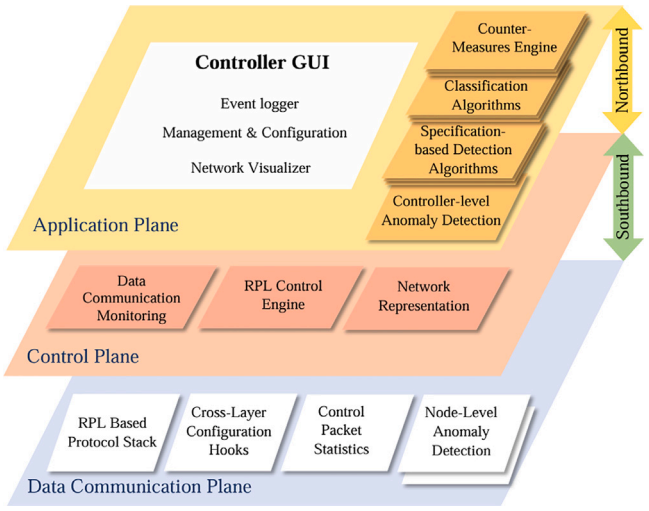


Fig. 7. ASSET architecture.

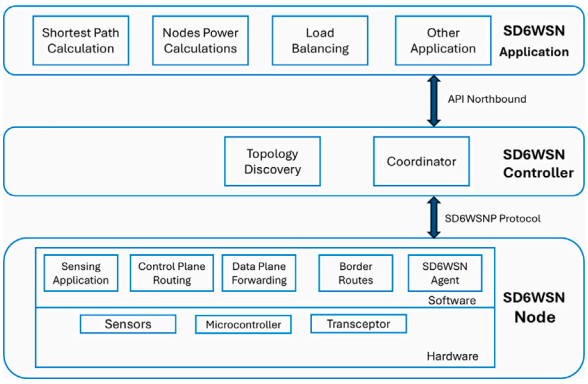


Fig. 8. SD6WSN architecture.

are exceeded. Essential Mode enhances this by gathering more detailed statistics to detect potential attacks and identify their nature, offering more proactive fault detection. Finally, Full-function Mode provides comprehensive monitoring, including node ranks and neighborhood information, allowing for an in-depth assessment of the network's condition.

This architecture defines an API that allows the controller to manage various aspects of network behavior. It can enable or disable the sending of ICMP statistics, control local/global repair mechanisms, disable the reset of the Trickle Timer, and add malicious nodes to a blacklist, preventing them from becoming parents in the RPL topology.

This architecture provides a robust solution to improve the security and reliability of RPL-based IoT networks, significantly reducing the vulnerabilities of some specific types of attacks.

SD6WSN

SD6WSN [91], proposed in 2018, introduces an SDN-based architecture specifically designed for 6LoWPAN Wireless Sensor Networks. The architecture focuses on addressing key limitations of existing IoT solutions, particularly the lack of detailed operational explanations and the challenges posed by the RPL protocol, especially in enabling efficient P2P communications.

The architecture leverages RPL as its routing protocol while introducing key enhancements to optimize its performance. A significant improvement is the ability to establish efficient P2P communication paths. Using the controller's global network view, SD6WSN dynamically calculates optimal routes between nodes, thereby reducing latency and energy consumption for more efficient data transfer. Fig. 8 illustrates this architecture.

To effectively manage the network, SD6WSN implements a custom SDN-based control protocol. This protocol minimizes communication overhead by sending updates to the controller only when significant changes occur in the network state, rather than at fixed intervals. This selective update mechanism conserves energy, reduces unnecessary control traffic, and improves scalability while maintaining responsiveness. SD6WSN is particularly beneficial for applications that require low latency, energy efficiency, and robust P2P communication.

SBR

SBR [92] is an SDN architecture for IoT proposed in 2021 to improve quality of service. It is based on an external controller and uses RPL in storing mode, where each node maintains its own routing table. The architecture uses the objective function SIGMA [93], which selects routes based on hop count and the standard deviation of ETX values to better reflect link stability. DAO messages are modified by reusing unused fields (option field) to include enhanced routing metrics, such as path length and path members.

The root node calculates SIGMA-ETX values and sends them back to the nodes through modified DAO-ACK messages, which carry this information as additional options. This allows for more precise routing decisions.

The SDN controller installs flow rules in the network nodes to meet QoS requirements. These rules, updated periodically, define conditions and corresponding actions (e.g., filtering, routing), helping to prevent congestion and enabling more efficient network management.

μSDN and its evolved versions

μSDN [94] is an SDN-based architecture specifically designed for resource-constrained IoT environments. It addresses the limitations of traditional SDN controllers, particularly their resource-intensive nature, which makes them unsuitable for lightweight IoT devices.

This architecture introduces a microcontroller-based SDN design, enabling the deployment of SDN functionality on devices with limited computational and memory capacities. Unlike conventional SDN controllers, μSDN embeds control capabilities directly into edge nodes, thereby distributing some control functions closer to the network's periphery. This hybrid approach combines centralized decision-making with localized execution, striking a balance between control efficiency and resource optimization.

Furthermore, μSDN employs a minimalist protocol for communication between the controller and IoT nodes, significantly reducing overhead. It defines the following control messages:

NSU (**N**ode **S**tate **U**pdate): This message carries state information of a node, such as connectivity status, to the controller and is sent periodically.

FTQ (**F**low **T**able **Q**uery): When a node needs to query the controller for routing information, it sends an FTQ message. The controller responds with the appropriate flow table entries for the node's forwarding decisions.

FTS (**F**low **T**able **S**et): This message is used by the controller to set or update the flow table entries at the nodes, dictating how packets should be forwarded based on the defined rules.

CONF (**C**onfiguration): This message configures network parameters or node-specific settings, such as updating routing metrics, according to the network's needs.

μSDN leverages RPL to handle the routing of control messages. Meanwhile, data routing decisions are calculated based on a lightweight heuristic algorithm designed to optimize energy efficiency while maintaining acceptable levels of latency and packet delivery.

EA-SDN/NFV [95] builds upon μSDN by integrating Network Function Virtualization (NFV), enabling the dynamic activation and deactivation of NFVs at relay nodes to optimize data aggregation. This architecture introduces an enhanced node design equipped with NFV capabilities, leveraging the synergy of SDN and NFV to improve network efficiency. To achieve optimal placement and minimize the number of active NFV nodes while reducing overall energy consumption, a heuristic algorithm (EA-SDN/NFV) is employed. This algorithm balances energy efficiency with network throughput, making it suitable for practical deployment. The architecture incorporates two key modules:

NMM (**N**FV **M**anagement **M**odule): Responsible for managing virtualized network functions, such as data aggregation, and providing an API for NFV-enabled nodes.

RMM (**R**oute **M**anagement **M**odule): Monitors the energy states of nodes to assist the controller in optimizing NFV activation and route assignment.

Regarding data aggregation, NFV nodes aggregate data packets when their buffers are full. This process involves collecting data from multiple source nodes and merging them into a single packet for transmission to the sink. By reducing the number of transmissions required, the aggregation process significantly enhances network efficiency. The controller orchestrates this process, selecting the nodes to perform aggregation based on the network's energy states and traffic patterns. This mechanism also minimizes packet loss and improves the packet delivery ratio, as fewer transmissions lead to reduced overhead and more reliable data delivery.

e-μSDN [96] is another enhanced version of μSDN, designed to improve adaptability and efficiency through dynamic RPL configuration. It introduces three key improvements over its predecessor. First, it reduces control overhead by sending NSU messages only when significant changes occur, avoiding unnecessary updates and conserving energy. Second, it proactively establishes routes ahead of data transmission, minimizing delays compared to μSDN's reactive approach. Third, it adapts RPL settings based on traffic patterns in smart buildings: using higher update frequencies during the day to handle dynamic topologies, and reducing control traffic at night when the network is more stable. This adaptive strategy achieves a balance between energy efficiency and network performance, ensuring optimal functionality regardless of external conditions.

Nx-IoT

Nx-IoT, proposed by Das et al. [97], is designed to address specific limitations of traditional IoT architectures, particularly their commercialization challenges within the industry. This architecture integrates the control layer directly into the network layer of the IoT framework, leveraging SDN principles to enhance network management and scalability.

A defining feature of Nx-IoT, as shown in Fig. 9, is its multi-controller design, which prioritizes both resilience and efficiency. Each switch connects to multiple controllers, ranked based on their proximity. The closest controller is designated as the primary, with others acting as backups. When the primary controller becomes overloaded, a secondary controller seamlessly takes over, reassigning tasks and notifying all switches to optimize future routing decisions.

This architecture utilizes OpenFlow as the communication protocol and RPL for routing, ensuring compatibility with resource-constrained environments typical of IoT. This combination of RPL and OpenFlow enables Nx-IoT to balance resource efficiency with robust control capabilities.

Following the exploration of various SDN architectures for LLNs, it is essential to classify and compare them according to their key characteristics. The following section provides a taxonomy of SDN architectures, highlighting the different design approaches and features that have been introduced to address the challenges of IoT networks. By categorizing these architectures, we can better understand their unique contributions and the evolution of SDN in the context of IoT.

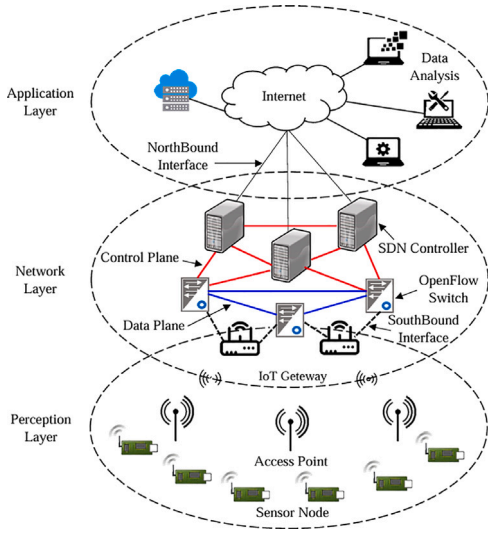


Fig. 9. Nx-IoT architecture.

4. Taxonomy of SDN architectures for IoT

The taxonomy, summarized in Table 6, presents a classification of SDN architectures for IoT based on key characteristics such as routing protocols, controllers, use cases, open-source availability, and more. This table serves as a valuable reference, providing a comprehensive overview of the different architectures and their unique features. By organizing these architectures according to their defining characteristics, it offers a clear comparison that helps to better understand their applicability and potential advantages for various IoT scenarios.

An analysis of this taxonomy reveals significant variations in their characteristics, reflecting a wide range of design strategies and priorities tailored to different IoT scenarios.

One of the main differentiators among these architectures is the choice of routing protocol. Architectures like CORAL SDN [85], MINOS [86], and e-μSDN [96] leverage the standardized RPL protocol, which is known for its low energy consumption and widespread adoption in IoT networks. This enables these architectures to integrate seamlessly with existing IoT setups, while also enhancing RPL's capabilities to address specific challenges such as mobility.

In contrast, architectures like SDN-WISE [78], STEER [80], and SSDWSN [81] opt for customized routing protocols that are specifically designed to optimize energy efficiency and address resource constraints more effectively. However, these tailored protocols may encounter challenges related to standardization and interoperability, potentially limiting their integration with existing IoT ecosystems.

Many architectures also rely on standardized APIs, such as OpenFlow [8] (e.g., SDN-WISE, STEER), to enhance compatibility with broader SDN ecosystems. However, some architectures adopt customized APIs to address specific needs, such as dynamic routing protocol adaptation or fault detection. Standardized APIs facilitate integration across diverse systems, while customized APIs enable the development of unique features suited to IoT contexts.

The choice and placement of controllers further distinguish these architectures. Some, like SD-WISE [79] and SSDWSN [81], rely on ONOS, a reference SDN platform widely used in research and industry [63], which provides scalability and reliability by leveraging well-established frameworks. Others, like CORAL SDN [85] and STEER [80], use customized controllers tailored to IoT-specific requirements. However, these custom controllers may lack compatibility with standards, making their integration into heterogeneous ecosystems more challenging.

The placement of the controller also varies. External controllers, commonly adopted in most architectures, offer centralized control

Table 6
Taxonomy of SDN architectures for IoT over LLNs.

Architecture	Year	Routing protocol	Controller	Southbound API	Aim	Use case	Availability
SDN-WISE [78]	2015	Customized	Customized - External	Standardized - OpenFlow	Improve flexibility and efficiency in sensor networks by enabling programmable stateful sensor nodes	Not specified	Yes [98]
TinySDN [82]	2015	Customized - CTP	Customized - Internal	Customized	Enable the deployment of several SDN controllers within the same wireless sensor network, offering better performances	Not specified	Yes [99]
SD-WISE [79]	2017	Customized	ONOS	Standardized - OpenFlow	Enhance SDN-WISE by reducing energy consumption, improving security, and providing greater flexibility	Not specified	Yes [100]
IT-SDN [84]	2017	Customized	Customized - External	Customized	Improve TinySDN by providing a more modular architecture, independent of the underlying operating system	Not specified	Yes [101]
CORAL SDN [85]	2017	Standardized - RPL	Customized - External	Customized	Enhance RPL for mobility and P2P communications support in dynamic IoT settings	Smart cities	Yes [102]
μSDN [94]	2018	Standardized - RPL	Customized - Internal	Customized	Using a micro-controller-based design to enable SDN on low-resource devices	Not specified	Yes [103]
SD6WSN [91]	2018	Customized	Customized - External	Customized	Enhance P2P communications and address other RPL protocol limitations through SDN	Not specified	Yes [104]
MINOS [86]	2019	Standardized - RPL	Customized - External	Customized	Support dynamic routing protocol selection to cater to diverse IoT service requirements	Smart cities	Yes [105]
VERO SDN [87]	2020	Customized	Customized - External	Customized	Reduce the overhead generated by the integration of SDN in resource-constrained environments	Smart cities	Yes [106]
EA-SDN/NFV [95]	2020	Standardized - RPL	Customized - Internal	Customized	Integrate NFV with SDN to provide data aggregation and improve energy efficiency in IoT networks	Not specified	Yes [107]
MIoT [88]	2020	Customized	Customized - External	Customized	Address mobility challenges while minimizing overhead, with a key innovation on intelligent mobility detection	High mobility IoT networks	Yes [108]
ASSET [90]	2021	Standardized - RPL	Customized - External	Customized	Address RPL vulnerabilities, such as battery-draining attacks and traffic spoofing, by detecting faults, identifying attackers, and implementing mitigation strategies	Networks vulnerable to RPL-targeted attacks	Yes [109]
SBR [92]	2021	Standardized - RPL	Customized - External	Customized	Enhance QoS: packet delivery, latency, and extend the lifetime of the network	Not specified	No
Nx-IoT [97]	2022	Standardized - RPL	Customized - External	Standardized - OpenFlow	Address commercialization challenges while using SDN in IoT networks by integrating the control layer into the network layer	Not specified	No
STEER [80]	2023	Customized	Customized - External	Standardized - OpenFlow	Integrate SDN with IDN to enable better network self-adaptation through an additional layer that maps intents to dynamic behaviors	Indoor environments	Yes [110]
DENIS [89]	2023	Customized	Customized - External	Customized	Optimize communication and minimize packet loss in high-density IoT networks	High-density networks	Yes [111]
e-μSDN [96]	2024	Standardized - RPL	Customized - Internal	Customized	Enhance adaptability and efficiency by allowing a dynamic RPL configuration along with traffic specificities	Smart buildings	No
SSDWSN [81]	2024	Customized	ONOS	Standardized - OpenFlow	integrate SDN into IoT while minimizing signaling, avoiding congestion, and extending network lifetime	Not specified	Yes [112]

and enhanced processing capabilities, but they may introduce latency and require robust communication links. In contrast, internal controllers, as seen in μSDN [94], TinySDN [82], and e-μSDN [96], enable SDN functionalities directly on constrained devices, reducing latency and communication overhead. While this approach offers significant advantages, it also presents challenges related to processing limitations [113].

The diversity of use cases further underscores the adaptability of SDN in addressing various IoT scenarios. Architectures like MIoT [88] and ASSET [90] tackle specialized challenges, such as mobility and RPL vulnerabilities, while others, such as DENIS [89], focus on optimizing communication in high-density networks. To better understand the focus of each architecture based on the specific challenges of LLNs discussed in Section 2.5, Table 7 highlights their respective contributions.

Building on this analysis, the following section discusses the limitations of these architectures and the key challenges that remain to be addressed.

5. Discussion

This study explores the integration of SDN into IoT over LLNs, a combination that, while ambitious, has successfully overcome many challenges that traditional approaches were unable to resolve. In recent years, SDN-based architectures for IoT have seen notable progress. For instance, the SSDWSN architecture proposes an interesting mechanism for reducing the overhead introduced by SDN. The use of Deep Reinforcement Learning in this context appears to be a promising solution to predict the network state and avoid frequent state updates, thereby significantly reducing SDN overhead.

Table 7
Contribution of SDN architectures in addressing LLNs challenges.

Architecture	QoS	Mobility	Security	Energy management
SDN-WISE [78]	✓	✗	✗	✗
TinySDN [82]	✓	✗	✗	✗
SD-WISE [79]	✓	✗	✓	✓
IT-SDN [84]	✓	✗	✗	✗
CORAL SDN [85]	✓	✓	✗	✗
μSDN [94]	✓	✗	✗	✗
SD6WSN [91]	✓	✗	✗	✗
MINOS [86]	✓	✓	✗	✗
VERO SDN [87]	✓	✓	✗	✗
EA-SDN/NFV [95]	✓	✗	✗	✓
MIoT [88]	✓	✓	✗	✗
ASSET [90]	✗	✗	✓	✗
SBR [92]	✓	✗	✗	✗
Nx-IoT [97]	✓	✗	✗	✗
STEER [80]	✗	✗	✗	✓
DENIS [89]	✓	✗	✗	✗
e-μSDN [96]	✓	✗	✗	✓
SSDWSN [81]	✓	✗	✗	✓

Such advancements offer encouraging signs regarding the feasibility of applying SDN in resource-constrained IoT environments. Nevertheless, despite these promising developments, several limitations remain that hinder widespread adoption. The remainder of this section discusses the key limitations and design considerations that must be addressed to ensure the successful deployment of SDN in IoT systems.

5.1. Limitations

Despite the growing number of studies focusing on the use of SDN in the IoT domain, its real-world adoption remains limited due to several technical and practical constraints. While most implementations are available as open source, which is a significant advantage, their documentation is often insufficient or incomplete. This shortcoming makes their deployment particularly challenging for researchers and engineers.

Moreover, another major obstacle lies in the fact that some of these architectures rely on outdated versions of development frameworks, and other compilation tools. These versions, which are sometimes obsolete or even no longer available in modern environments, create significant challenges in terms of installation and compatibility. This greatly complicates their use and integration into contemporary infrastructures.

In addition, all these architectures were developed using Contiki OS [114], an open-source operating system designed for WSNs and IoT devices, with the exception of TinySDN, which was originally implemented on TinyOS [83]. TinyOS, while historically significant in the WSN community, presents several limitations. It is written in nesC, a language that is less accessible to many developers, and its programming model is often considered restrictive and complex. Additionally, TinyOS offers limited support for modern protocols and lacks the modularity and flexibility required for evolving IoT applications.

In contrast, Contiki provides a more modern and flexible development environment, better suited to resource-constrained devices, such as battery-powered sensors. It supports multiple low-energy communication protocols and features a lightweight, event-driven programming model that is easier to adopt. Most of these architectures were built on Contiki 3.0, an older version of the system. Only the μSDN architecture [115] has an implementation based on Contiki-NG [116], the latest version of Contiki, which offers significant improvements in performance, compatibility, and support for modern technologies.

Relying on Contiki 3.0 instead of Contiki-NG has several drawbacks. First, Contiki 3.0 is no longer maintained, making it vulnerable to security issues and less adaptable to current technological advancements. Additionally, key features such as modern protocol management and performance optimization are better supported in Contiki-NG [117]. Lastly, using outdated versions complicates integration with recent tools and limits collaboration opportunities with research and development communities that rely on more up-to-date platforms.

In terms of performance, most of these architectures have been evaluated using Cooja, a simulator specifically designed for Contiki OS. While Cooja is a powerful tool for testing and experimentation, it introduces certain limitations that hinder the practical adoption of these architectures. Simulations, by nature, do not fully replicate real-world conditions, such as unpredictable interference, hardware-specific behaviors, and the complexity of large-scale deployments. Consequently, results obtained through Cooja may not accurately reflect the performance of these systems in real-life scenarios, making it challenging to assess their reliability and scalability when deployed in real environments.

Moreover, comparing the performance of the proposed solutions with basic RPL (without SDN) is essential to evaluate the trade-offs involved, particularly in terms of overhead and energy consumption. Although SDN can improve network performance, such as QoS and security, its adoption must be carefully weighed against the additional cost it introduces. If the control overhead is too high, the energy consumption of resource-constrained IoT nodes may increase significantly, potentially shortening the network lifetime. In such cases, the benefits of SDN may not justify its deployment, especially in LLNs where energy efficiency is a primary concern. Therefore, it is crucial to strike a balance between performance gains and energy sustainability.

One of the key causes of this overhead is the use of specialized control messages combined with the extreme centralization of the control plane. To mitigate this, hybrid approaches such as Whisper have been proposed [118]. Rather than introducing new messages, Whisper enables centralized control by sending instructions that are compatible with existing distributed protocols, such as 6TiSCH [71]. This approach reduces overhead, improves energy efficiency, and limits performance degradation. While promising, Whisper remains limited to industrial environments [119], and has not yet been explored or adapted for general public IoT deployments.

However, most existing studies fall short in adopting such effective mechanisms to mitigate SDN control overhead. In addition, their evaluations are often limited, lacking long-term assessments of energy consumption and performance degradation. This raises concerns about the real viability of these architectures in practical IoT scenarios.

To better illustrate and concretize these limitations and link them with the architectures in Section 3, Table 8 summarizes the version of the operating system used, the evaluation tool employed (simulator or real platform) and whether a comparison with respect to basic RPL was considered.

Table 8
Evaluation setups of SDN architectures for LLNs.

Architecture	OS version	Evaluation method		RPL benchmarking?
		Simulator	Testbed	
SDN-WISE [78]	Contiki 3.0	✓	✓	No
TinySDN [82]	not specified	✓	✗	No
SD-WISE [79]	Contiki 3.0	✗	✓	No
IT-SDN [84]	Contiki 3.0	✓	✗	Yes
CORAL SDN [85]	Contiki 3.0	✓	✗	No
μSDN [94]	Contiki 3.0 & Contiki-ng	✓	✗	Yes
SD6WSN [91]	Contiki 3.0	✓	✗	Yes
MINOS [86]	Contiki 3.0	✓	✗	No
VERO SDN [87]	Contiki 3.0	✓	✗	No
EA-SDN/NFV [95]	Contiki 3.0	✓	✗	No
MIoT [88]	Contiki 3.0	✓	✗	No
ASSET [90]	Contiki 3.0	✓	✗	No
SBR [92]	not specified	✓	✗	No
Nx-IoT [97]	not specified	✗	✓	No
STEER [80]	Contiki 3.0	✓	✗	No
DENIS [89]	Contiki 3.0	✓	✗	No
e-μSDN [96]	Contiki 3.0	✓	✗	Yes
SSDWSN [81]	Contiki 3.0	✓	✗	No

5.2. Future challenges

The successful adoption of SDN-based architectures for IoT systems faces several critical challenges. To overcome these limitations, it is essential to design architectures with the following key considerations:

- **Architecture Design:**

The design of SDN architectures for LLNs should consider several key characteristics.

One crucial aspect is the choice of the operating system, as it directly impacts performance, scalability, and security. Contiki OS [114] has been widely adopted in SDN-based IoT architectures due to its lightweight design and efficiency in resource-constrained environments. However, its latest version, Contiki-NG [116], introduces significant improvements over older versions like Contiki 3.0, including enhanced hardware support, stronger security features, and new communication protocols. These advances are essential to ensure scalability and robust security in IoT networks. By adopting the latest version of Contiki OS, architectures can take advantage of these enhancements, ensuring compatibility with modern IoT hardware and software while optimizing network performance and security.

In addition, the use of standardized protocols is essential to ensure seamless interoperability between devices, networks, and layers within an SDN architecture. When designing the system, using protocols such as RPL and OpenFlow for routing and communication facilitates integration across different platforms, allowing devices to interact efficiently without compatibility issues.

Beyond standardized protocols, a modular SDN controller design further enhances flexibility by enabling the seamless integration of new services without disrupting the overall network structure. This modular approach allows for easy addition of new applications or functionalities (e.g., traffic management, security protocols) without requiring major modifications to the existing infrastructure. Such adaptability is particularly valuable in IoT environments, where new use cases constantly emerge. By ensuring compatibility with the latest innovations, a modular design helps maintain the relevance and efficiency of the SDN architecture over time.

Finally, ensuring resilience against failures and attacks is a critical aspect of SDN architecture design. The failure or targeted attacks on the SDN controller pose significant risks, as they can disrupt the entire network. To mitigate this, it is essential to implement mechanisms that enable the network to continue operating, either through the use of a secondary controller or through a mechanism that allows the network to operate in a basic state without the controller. In this case, such resilience ensures that essential functionalities, like routing and communication, are maintained until the controller is restored, thereby enhancing the overall reliability and robustness of the system.

- **Performance Analysis:**

Performance evaluation of SDN integration for the IoT should be performed on real-world platforms, rather than relying solely on simulations. Realistic scenarios with reproducible results are essential to validate the practicality and effectiveness of the proposed solutions.

In addition, analyzing the overhead in an SDN-based IoT network is essential to assess its efficiency compared to traditional architectures. The impact of SDN on node resources must be evaluated to ensure that SDN does not introduce excessive overhead, making it less attractive. Furthermore, evaluating how this overhead evolves as the network scales is necessary, as a centralized SDN approach may become a bottleneck in large deployments. Testing different strategies, such as optimization techniques to reduce control traffic, could help find a balance between performance and overhead.

- **Community Collaboration:**

Open-source and well-documented architectures foster a strong community around the project, driving innovation, diagnosis, and the development of new features. This community-driven approach ensures that the network can continuously evolve to meet the latest security threats and technological advancements. Regular updates are also essential to maintain the system's functionality, reliability, and relevance, allowing it to stay aligned with emerging challenges and benefit from ongoing contributions from the global development community.

6. Conclusion

This work explores the integration of SDN into IoT over LLNs for non-industrial applications, emphasizing the necessity of such an approach. We presented a comprehensive analysis of state-of-the-art research, reviewing a wide range of studies, highlighting their key contributions, and organizing existing SDN architectures for IoT over LLNs. Additionally, we proposed an up-to-date taxonomy of these architectures, offering a clear classification to enhance the understanding of their design principles and focus areas.

As a supplementary contribution, our study not only sheds light on the latest advancements in SDN-based IoT architectures but also identifies critical gaps and limitations that hinder their adoption. Numerous

architectures have been designed, each addressing specific challenges of LLNs, such as quality of service, mobility, or security. The most recent ones show promising progress and significant advancements, offering solutions to the challenges that hinder this integration. However, despite these targeted efforts, none have yet achieved widespread deployment. This is due to persistent shortcomings, including difficulties in installation and deployment, inconsistent performance evaluations, and challenges in seamlessly integrating with existing IoT infrastructures. While SDN-IoT integration holds significant potential to enhance LLNs, overcoming these barriers is essential to transition from academic research to real-world applications.

To bridge this gap, we have identified key considerations that must be taken into account when designing SDN architectures for IoT. Addressing these aspects is crucial for mitigating current limitations and enabling SDN-based solutions to be effectively deployed in real-world IoT environments, fully leveraging the potential of IoT.

CRediT authorship contribution statement

Liticia Djennadi: Writing – original draft, Visualization. **Gladys Diaz:** Supervision. **Khaled Boussetta:** Supervision. **Christophe Cerin:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] S. Madakam, R. Ramaswamy, S. Tripathi, Internet of things (IoT): A literature review, J. Comput. Commun. 3 (5) (2015) 164–173, http://dx.doi.org/10.4236/jcc.2015.35021.

[2] H. Rehan, Internet of things (IoT) in smart cities: Enhancing urban living through technology, J. Eng. Technol. 5 (1) (2023) 1–16.

[3] H. Yang, W. Lee, H. Lee, IoT smart home adoption: the importance of proper level automation, J. Sensors 2018 (1) (2018) 6464036, http://dx.doi.org/10.1155/2018/6464036.

[4] S. Nižetić, P. Šolić, D.L.-d.-I. Gonzalez-De, L. Patrono, et al., Internet of things (IoT): Opportunities, issues and challenges towards a smart and sustainable future, J. Clean. Prod. 274 (2020) 122877, http://dx.doi.org/10.1016/j.jclepro.2020.122877.

[5] Á.L. Valdivieso Caraguay, A. Benito Peral, L.I. Barona Lopez, L.J. Garcia Villalba, SDN: Evolution and opportunities in the development IoT applications, Int. J. Distrib. Sens. Networks 10 (5) (2014) 735142, http://dx.doi.org/10.1155/2014/7351.

[6] O.A. A., E. Seun, A.A. O., O.F. Y., Introduction to software defined networks (SDN), Int. J. Appl. Inf. Syst. 11 (7) (2016) 10–14, http://dx.doi.org/10.5120/ijais2016451623.

[7] N. Anerousis, P. Chemouil, A.A. Lazar, N. Mihai, S.B. Weinstein, The origin and evolution of open programmable networks and SDN, IEEE Commun. Surv. & Tutorials 23 (3) (2021) 1956–1971, http://dx.doi.org/10.1109/COMST.2021.3060582.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74, http://dx.doi.org/10.1145/1355734.1355746.

[9] B.A.A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Commun. Surv. & Tutorials 16 (3) (2014) 1617–1634, http://dx.doi.org/10.1109/SURV.2014.012214.00180.

[10] T. Ninikrishna, S. Sarkar, R. Tengshe, M.K. Jha, L. Sharma, V. Daliya, S.K. Routray, Software defined IoT: Issues and challenges, in: 2017 International Conference on Computing Methodologies and Communication, ICCMC, IEEE, 2017, pp. 723–726, http://dx.doi.org/10.1109/ICCMC.2017.8282560.

[11] W.Z. Khan, M. Rehman, H.M. Zangoti, M.K. Afzal, N. Armi, K. Salah, Industrial internet of things: Recent advances, enabling technologies and open challenges, Comput. Electr. Eng. 81 (2020) 106522, http://dx.doi.org/10.1016/j.compeleceng.2019.106522.

[12] N.N. Josbert, M. Wei, W. Ping, A. Rafiq, A look into smart factory for industrial IoT driven by SDN technology: A comprehensive survey of taxonomy, architectures, issues and future research orientations, J. King Saud University- Comput. Inf. Sci. (2024) 102069, http://dx.doi.org/10.1016/j.jksuci.2024.102069.

[13] N. Bizanis, F.A. Kuipers, SDN and virtualization solutions for the internet of things: A survey, IEEE Access 4 (2016) 5591–5606, http://dx.doi.org/10.1109/ACCESS.2016.2607786.

[14] O. Salman, I. Elhajj, A. Chehab, A. Kayssi, IoT survey: An SDN and fog computing perspective, Comput. Netw. 143 (2018) 221–246, http://dx.doi.org/10.1016/j.comnet.2018.07.020.

[15] H. Zemrane, Y. Baddi, A. Hasbi, SDN-based solutions to improve IoT: survey, in: 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), IEEE, 2018, pp. 588–593, http://dx.doi.org/10.1109/CIST.2018.8596577.

[16] I. Alam, K. Sharif, F. Li, Z. Latif, M.M. Karim, S. Biswas, B. Nour, Y. Wang, A survey of network virtualization techniques for internet of things using SDN and NFV, ACM Comput. Surv. 53 (2) (2020) 1–40, http://dx.doi.org/10.1145/3379444.

[17] P.P. Ray, N. Kumar, SDN/NFV architectures for edge-cloud oriented IoT: A systematic review, Comput. Commun. 169 (2021) 129–153, http://dx.doi.org/10.1016/j.comcom.2021.01.018.

[18] S. Siddiqui, S. Hameed, S.A. Shah, I. Ahmad, A. Aneiba, D. Draheim, S. Dustdar, Toward software-defined networking-based IoT frameworks: A systematic literature review, taxonomy, open challenges and prospects, IEEE Access 10 (2022) 70850–70901, http://dx.doi.org/10.1109/ACCESS.2022.3188311.

[19] K.H. Manguri, S.M. Omer, SDN for IoT environment: a survey and research challenges, in: ITM Web of Conferences, 42, EDP Sciences, 2022, p. 01005, http://dx.doi.org/10.1051/itmconf/20224201005.

[20] S. Javanmardi, M. Shojafar, R. Mohammadi, M. Alazab, A.M. Caruso, An SDN perspective IoT-fog security: A survey, Comput. Netw. 229 (2023) 109732, http://dx.doi.org/10.1016/j.comnet.2023.109732.

[21] S. Shafiq, M.S. Rahman, S.A. Shaon, I. Mahmud, A.S. Hosen, A review on software-defined networking for internet of things inclusive of distributed computing, blockchain, and mobile network technology: Basics, trends, challenges, and future research potentials, Int. J. Distrib. Sens. Networks 2024 (1) (2024) 9006405, http://dx.doi.org/10.1155/2024/9006405.

[22] A.A. Laghari, K. Wu, R.A. Laghari, M. Ali, A.A. Khan, A review and state of art of internet of things (IoT), Arch. Comput. Methods Eng. (2021) 1–19.

[23] A.A. Alli, M.M. Alam, The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications, Internet Things 9 (2020) 100177, http://dx.doi.org/10.1016/j.iot.2020.100177.

[24] J. Ding, M. Nemati, C. Ranaweera, J. Choi, IoT connectivity technologies and applications: A survey, IEEE Access 8 (2020) 67646–67673, http://dx.doi.org/10.1109/ACCESS.2020.2985932.

[25] J. Fürst, K. Chen, H.-S. Kim, P. Bonnet, Evaluating bluetooth low energy for IoT, in: 2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench), IEEE, 2018, pp. 1–6, http://dx.doi.org/10.1109/CPSBench.2018.00007.

[26] H. Zou, Y. Zhou, J. Yang, C.J. Spanos, Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT, Energy Build. 174 (2018) 309–322, http://dx.doi.org/10.1016/j.enbuild.2018.06.040.

[27] H.-Y. Chang, A connectivity-increasing mechanism of ZigBee-based IoT devices for wireless multimedia sensor networks, Multimedia Tools Appl. 78 (5) (2019) 5137–5154, http://dx.doi.org/10.1109/EUC.2008.71.

[28] M.A. Al-Absi, A.A. Al-Absi, M. Sain, H.J. Lee, A state of the art: future possibility of 5G with IoT and other challenges, Smart Heal. Anal. IoT Enabled Environ. (2020) 35–65, http://dx.doi.org/10.1007/978-3-030-37551-5_3.

[29] M. Jouhari, N. Saeed, M.-S. Alouini, E.M. Amhoud, A survey on scalable LoRaWAN for massive IoT: Recent advances, potentials, and challenges, IEEE Commun. Surv. & Tutorials 25 (3) (2023) 1841–1876, http://dx.doi.org/10.1109/COMST.2023.3274934.

[30] K. Kim, G. Montenegro, S. Park, I. Chakeres, C. Perkins, Dynamic MANET on-demand for 6LoWPAN (DYMO-low) routing, Internet Eng. Task Force, Internet-Draft. (2007).

[31] K. Kim, 6LoWPAN ad hoc on-demand distance vector routing (LOAD), Draft-Daniel- 6lowpan- Load- Adhoc- Routing- 02. Txt (2007).

[32] S. Dawson-Haggerty, A. Tavakoli, D. Culler, Hydro: A hybrid routing protocol for low-power and lossy networks, in: 2010 First IEEE International Conference on Smart Grid Communications, IEEE, 2010, pp. 268–273, http://dx.doi.org/10.1109/SMARTGRID.2010.5622053.

[33] T. Watteyne, Routing requirements for urban low-power and lossy networks, 2009, http://Datatracker.Ietf.Org/Doc/Rfc5548/.

[34] A. Brandt, J. Buron, G. Porcu, Home Automation Routing Requirements in Low-Power and Lossy Networks, Tech. rep., 2010.

[35] K. Pister, P. Thubert, S. Dwars, T. Phinney, Industrial Routing Requirements in Low-Power and Lossy Networks, Tech. rep., 2009.

[36] J.V. Sobral, J.J. Rodrigues, R.A. Rabêlo, J. Al-Muhtadi, V. Korotaev, Routing protocols for low power and lossy networks in internet of things applications, Sensors 19 (9) (2019) 2144, http://dx.doi.org/10.3390/s19092144.

[37] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, Tech. rep., 2012.

[38] I. Unwala, Z. Taqvi, J. Lu, Thread: An iot protocol, in: 2018 IEEE Green Technologies Conference (GreenTech), IEEE, 2018, pp. 161–167, http://dx.doi.org/10.1109/GreenTech.2018.00037.

[39] P. Levis, T. Clausen, J. Hui, O. Gnawali, J. Ko, RFC 6206: The trickle algorithm, 2011.

[40] K.A. Darabkh, M. Al-Akhras, RPL over internet of things: Challenges, solutions, and recommendations, in: 2021 IEEE International Conference on Mobile Networks and Wireless Communications, ICMNWC, 2021, pp. 1–7, http://dx.doi.org/10.1109/ICMNWC52512.2021.9688375.

[41] P. Arivubrakan, G. Kanagachidambaresan, K-trickle: performance evaluation and impact on quality of service in resource-constrained networks, Int. J. Data Sci. Anal. (2024) 1–10, http://dx.doi.org/10.1007/s41060-024-00531-y.

[42] S.-T. Liu, S.-D. Wang, Improved trickle algorithm toward low power and better route for the RPL routing protocol, IEEE Access 10 (2022) 83322–83335, http://dx.doi.org/10.1109/ACCESS.2022.3196693.

[43] A. Verma, V. Ranga, Security of RPL based 6lowpan networks in the internet of things: A review, IEEE Sensors J. 20 (11) (2020) 5666–5690, http://dx.doi.org/10.36227/techrxiv.12121971.

[44] T.G.B. Kala Venugopal, A combined metric objective function for RPL load balancing in internet of things, Int. J. Internet Things 10 (1) (2022) 22–31.

[45] A.F. Bamogo, P. Poda, Optimizing RPL routing by a multi-metric combination, EasyChair Prepr. (4759) (2020) 1–11.

[46] H. Echoukairi, A. Ouacha, J. Oubaha, M. El Ghmary, A new objective function for RPL based on combined metrics in mobile IoT, J. Commun. 18 (5) (2023) 301–309, http://dx.doi.org/10.12720/jcm.18.5.301-309.

[47] S. Sarwar, S. Rauf, R. Rasheed, L. Aslam, Energy-aware routing in internet of things (IoT), in: 2019 2nd International Conference on Communication, Computing and Digital Systems (C-CODE), 2019, pp. 81–86, http://dx.doi.org/10.1109/C-CODE.2019.8680974.

[48] S. Sankar, P. Srinivasan, Energy and load aware routing protocol for internet of things, Int. J. Adv. Appl. Sci. (IJAAS) 7 (3) (2018) 255–264, http://dx.doi.org/10.11591/ijaas.v7.i3.pp255-264.

[49] A. Seyfollahi, M. Mainuddin, T. Taami, A. Ghaffari, RM-RPL: reliable mobility management framework for RPL-based IoT systems, Clust. Comput. 27 (4) (2024) 4449–4468, http://dx.doi.org/10.1007/s10586-023-04199-0.

[50] F. Kaviani, M. Soltanaghaei, CQARPL: Congestion and qos-aware RPL for IoT applications under heavy traffic, J. Supercomput. 78 (14) (2022) 16136–16166, http://dx.doi.org/10.1007/s11227-022-04488-2.

[51] H. Lamaazi, N. Benamar, RPL enhancement based FL-trickle: A novel flexible trickle algorithm for low power and lossy networks, Wirel. Pers. Commun. 110 (3) (2020) 1403–1428, http://dx.doi.org/10.1007/s11277-019-06792-2.

[52] S.-T. Liu, S.-D. Wang, Improved trickle algorithm toward low power and better route for the RPL routing protocol, IEEE Access 10 (2022) 83322–83335, http://dx.doi.org/10.1109/ACCESS.2022.3196693.

[53] M. Vučinić, M. Król, B. Jonglez, T. Coladon, B. Tourancheau, Trickle-D: High fairness and low transmission load with dynamic redundancy, IEEE Internet Things J. 4 (5) (2017) 1477–1488, http://dx.doi.org/10.1109/JIOT.2017.2650318.

[54] M.B. Yassein, S. Aljawarneh, et al., A new elastic trickle timer algorithm for internet of things, J. Netw. Comput. Appl. 89 (2017) 38–47, http://dx.doi.org/10.1016/j.jnca.2017.01.024.

[55] A. Aghaei, J.A. Torkestani, H. Kermajani, A. Karimi, LA-trickle: A novel algorithm to reduce the convergence time of the wireless sensor networks, Comput. Netw. 196 (2021) 108241, http://dx.doi.org/10.1016/j.comnet.2021.108241.

[56] S. Hoghooghi, R. Javidan, Proposing a new method for improving RPL to support mobility in the internet of things, IET Networks 9 (2) (2020) 48–55, http://dx.doi.org/10.1049/iet-net.2019.0152.

[57] K.S. Bhandari, I.-H. Ra, G. Cho, Multi-topology based qos-differentiation in RPL for internet of things applications, IEEE Access 8 (2020) 96686–96705, http://dx.doi.org/10.1109/ACCESS.2020.2995794.

[58] A. Verma, V. Ranga, CoSec-RPL: detection of copycat attacks in RPL based 6LoWPANs using outlier analysis, Telecommun. Syst. 75 (1) (2020) 43–61, http://dx.doi.org/10.1007/s11235-020-00674-w.

[59] A. Mayzaud, R. Badonnel, I. Chrisment, A distributed monitoring strategy for detecting version number attacks in RPL-based networks, IEEE Trans. Netw. Serv. Manag. 14 (2) (2017) 472–486, http://dx.doi.org/10.1109/TNSM.2017.2705290.

[60] L. Zhu, M.M. Karim, K. Sharif, C. Xu, F. Li, X. Du, M. Guizani, SDN controllers: A comprehensive analysis and performance evaluation study, ACM Comput. Surv. 53 (6) (2020) 1–40, http://dx.doi.org/10.1145/3421764.

[61] A. Eftimie, E. Borcoci, SDN controller implementation using OpenDaylight: experiments, in: 2020 13th International Conference on Communications, COMM, IEEE, 2020, pp. 477–481, http://dx.doi.org/10.1109/COMM48946.2020.9142044.

[62] D.J. Hamad, K.G. Yalda, N. ÈšÄfpuÈ, Performance assessment of QoS metrics in software defined networking using floodlight controller, JOIV: Int. J. Informatics Vis. 7 (3) (2023) 631–637, http://dx.doi.org/10.30630/joiv.7.3.1288.

[63] L.V. Ruchel, R.C. Turchetti, E.T. de Camargo, Evaluation of the robustness of SDN controllers ONOS and ODL, Comput. Netw. 219 (2022) 109403, http://dx.doi.org/10.1016/j.comnet.2022.109403.

[64] S. Bhardwaj, S.N. Panda, Performance evaluation using RYU SDN controller in software-defined networking environment, Wirel. Pers. Commun. 122 (1) (2022) 701–723, http://dx.doi.org/10.29304/jqcm.2022.14.1.879.

[65] Y. Tateiwa, A. Asano, Y. Kim, Y. Katayama, M. Niimura, Proposal of an event visualization system for debugging in software-defined networking exercises using trema, in: 2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan), IEEE, 2020, pp. 1–2, http://dx.doi.org/10.1109/ICCE-Taiwan49838.2020.9258317.

[66] A. Golmohammadi, M. Zhang, A. Arcuri, Testing restful apis: A survey, ACM Trans. Softw. Eng. Methodol. 33 (1) (2023) 1–41, http://dx.doi.org/10.48550/arXiv.2212.14604.

[67] Z. Yang, Y. Cui, B. Li, Y. Liu, Y. Xu, Software-defined wide area network (SD-WAN): Architecture, advances and opportunities, in: 2019 28th International Conference on Computer Communication and Networks, ICCCN, IEEE, 2019, pp. 1–9, http://dx.doi.org/10.1109/ICCCN.2019.8847124.

[68] A.A. Barakabitze, A. Ahmad, R. Mijumbi, A. Hines, 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges, Comput. Netw. 167 (2020) 106984, http://dx.doi.org/10.48550/arXiv.1912.02802.

[69] J.d.C. Silva, P.H. Pereira, L.L. de Souza, C.N. Marins, G.A. Marcondes, J.J. Rodrigues, Performance evaluation of IoT network management platforms, in: 2018 International Conference on Advances in Computing, Communications and Informatics, ICACCI, IEEE, 2018, pp. 259–265, http://dx.doi.org/10.1016/j.comcom.2022.01.010.

[70] I. Ahammad, M.A.R. Khan, Z.U. Salehin, Qos performance enhancement policy through combining fog and SDN, Simul. Model. Pr. Theory 109 (2021) 102292, http://dx.doi.org/10.1016/j.simpat.2021.102292.

[71] F.A. Alaba, Network protocols for the internet of things, in: Internet of Things: A Case Study in Africa, Springer, 2024, pp. 115–129.

[72] Z.E. Ahmed, A.A. Hashim, R.A. Saeed, M.M. Saeed, Mobility management enhancement in smart cities using software defined networks, Sci. Afr. 22 (2023) e01932, http://dx.doi.org/10.1016/j.sciaf.2023.e01932.

[73] M.M. Salim, S. Rathore, J.H. Park, Distributed denial of service attacks and its defenses in IoT: a survey, J. Supercomput. 76 (2020) 5320–5363, http://dx.doi.org/10.48550/arXiv.2008.01345.

[74] A. Al Hayajneh, M.Z.A. Bhuiyan, I. McAndrew, Improving internet of things (IoT) security with software-defined networking (SDN), Computers 9 (1) (2020) 8, http://dx.doi.org/10.1109/GLOCOM.2016.7841889.

[75] A. Gasouma, K.M. Yusof, A. Mubarakali, O.E. Tayfour, Software defined network for energy efficiency in IoT and RPL networks, Soft Comput. (2023) 1–10, http://dx.doi.org/10.1007/s00500-023-08608-9.

[76] S. Yousefi, H. Karimipour, F. Derakhshan, Data aggregation mechanisms on the internet of things: a systematic literature review, Internet Things 15 (2021) 100427, http://dx.doi.org/10.1016/j.iot.2021.100427.

[77] Y.-B. Lin, S.-Y. Wang, C.-C. Huang, C.-M. Wu, The SDN approach for the aggregation/disaggregation of sensor data, Sensors 18 (7) (2018) 2025, http://dx.doi.org/10.3390/s18072025.

[78] L. Galluccio, S. Milardo, G. Morabito, S. Palazzo, SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks, in: 2015 IEEE Conference on Computer Communications, INFOCOM, IEEE, 2015, pp. 513–521, http://dx.doi.org/10.1109/INFOCOM.2015.7218418.

[79] A.-C.G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, S. Palazzo, SD-WISE: A software-defined wireless sensor network, 2017, http://dx.doi.org/10.1016/j.comnet.2019.04.029, arXiv preprint arXiv:1710.09147.

[80] B.M. Cordeiro, R. Rodrigues Filho, I.G. Júnior, F.M. Costa, STEER: An architecture to support self-adaptive IoT networks for indoor monitoring applications, J. Internet Serv. Appl. 14 (1) (2023) 107–123, http://dx.doi.org/10.5753/jisa.2023.3084.

[81] M. Alsaeedi, M.M. Mohamad, A. Al-Roubaiey, SSDWSN: A scalable software-defined wireless sensor networks, IEEE Access (2024) http://dx.doi.org/10.1109/ACCESS.2024.3362353.

[82] B.T. De Oliveira, L.B. Gabriel, C.B. Margi, TinySDN: Enabling multiple controllers for software-defined wireless sensor networks, IEEE Lat. Am. Trans. 13 (11) (2015) 3690–3696, http://dx.doi.org/10.1016/j.compeleceng.2017.02.026.

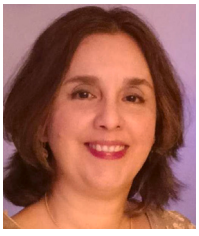
[83] M. Amjad, M. Sharif, M.K. Afzal, S.W. Kim, TinyOS-new trends, comparative views, and supported sensing applications: A review, IEEE Sensors J. 16 (9) (2016) 2865–2889, http://dx.doi.org/10.1109/JSEN.2016.2519924.

[84] R.C.A. Alves, D.A.G.d. Oliveira, G.A. Núñez Segura, C.B. Margi, It-sdn: Improved architecture for sdwsn, Anais (2017).

[85] T. Theodorou, L. Mamatas, CORAL-SDN: A software-defined networking solution for the internet of things, in: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2017, pp. 1–2, http://dx.doi.org/10.1109/NFV-SDN.2017.8169870.

[86] T. Theodorou, G. Violettas, P. Valsamas, S. Petridou, L. Mamatas, A multi-protocol software-defined networking solution for the internet of things, IEEE Commun. Mag. 57 (10) (2019) 42–48, http://dx.doi.org/10.1109/MCOM.001.1900056.
[87] T. Theodorou, L. Mamatas, A versatile out-of-band software-defined networking solution for the internet of things, IEEE Access 8 (2020) 103710–103733, http://dx.doi.org/10.1109/ACCESS.2020.2999087.
[88] T. Theodorou, L. Mamatas, SD-MIoT: A software-defined networking solution for mobile internet of things, IEEE Internet Things J. 8 (6) (2020) 4604–4617, http://dx.doi.org/10.1109/JIOT.2020.3027427.
[89] T. Theodorou, L. Mamatas, DENIS-SDN: Software-defined network slicing solution for dense and ultra-dense IoT networks, 2023, http://dx.doi.org/10.48550/arXiv.2312.13662, arXiv preprint arXiv:2312.13662.
[90] G. Violettas, G. Simoglou, S. Petridou, L. Mamatas, A softwarized intrusion detection system for the RPL-based internet of things networks, Future Gener. Comput. Syst. 125 (2021) 698–714, http://dx.doi.org/10.1016/j.future.2021.07.013.
[91] M.L. Miguel, E. Jamhour, M.E. Pellenz, M.C. Penna, SDN architecture for 6LoWPAN wireless sensor networks, Sensors 18 (11) (2018) 3738, http://dx.doi.org/10.3390/s18113738.
[92] P. Sanmartin, K. Avila, S. Valle, J. Gomez, D. Jabba, SBR: A novel architecture of software defined network using the RPL protocol for internet of things, IEEE Access 9 (2021) 119977–119986, http://dx.doi.org/10.1109/ACCESS.2021.3106950.
[93] P. Sanmartin, A. Rojas, L. Fernandez, K. Avila, D. Jabba, S. Valle, Sigma routing metric for RPL protocol, Sensors 18 (4) (2018) 1277, http://dx.doi.org/10.3390/s18041277.
[94] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, D. Simeonidou, Evolving SDN for low-power IoT networks, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), IEEE, 2018, pp. 71–79, http://dx.doi.org/10.1109/NETSOFT.2018.8460125.
[95] D. Saha, M. Shojaee, M. Baddeley, I. Haque, An energy-aware SDN/NFV architecture for the internet of things, in: 2020 IFIP Networking Conference (Networking), 2020, pp. 604–608.
[96] L. Djennadi, G. Diaz, K. Boussetta, C. Cerin, SDN-based approach for adaptive reconfiguration of routing in IoT for smart-buildings, in: 2024 IEEE 25th International Conference on High Performance Switching and Routing, HPSR, IEEE, 2024, pp. 137–142, http://dx.doi.org/10.1109/HPSR62440.2024.10635953.
[97] R.K. Das, N. Ahmed, A.K. Maji, G. Saha, Nx-iot: Improvement of conventional iot framework by incorporating sdn infrastructure, IEEE Internet Things J. 10 (3) (2022) 2473–2482, http://dx.doi.org/10.1109/jiot.2022.3215650.
[98] SDN-WISE, 2025, URL https://github.com/sdnwiselab/sdnwiselab.github.io, (Accessed 6 March 2025).
[99] Tinysdn, 2025, URL https://github.com/batmancn/TinySDNController, (Accessed 13 May 2025).
[100] SD-WISE, 2025, URL https://github.com/sdnwiselab/onos, (Accessed 6 March 2025).
[101] IT-SDN, 2025, URL https://github.com/cintiabmargi/IT-SDN_dowloads, (Accessed 13 May 2025).
[102] Coral SDN, 2025, URL https://github.com/SWNRG/coral-sdnm (Accessed 6 March 2025).
[103] μ SDN, 2025, URL https://github.com/mbaddeley/usdn, (Accessed 6 March 2025).
[104] SD6WSN, 2025, URL https://github.com/marciolm/sd6wsn, (Accessed 6 March 2025).
[105] MINOS, 2025, URL https://github.com/SWNRG/minos, (Accessed 6 March 2025).
[106] VERO SDN, 2025, URL https://github.com/SWNRG/vero-sdn, (Accessed 6 March 2025).
[107] EA-SDN/NFV, 2025, URL https://github.com/dipon76/EA-SDN-NFV-THESIS-final, (Accessed 6 March 2025).
[108] MIoT, 2025, URL https://github.com/SWNRG/SD-MIoT, (Accessed 6 March 2025).
[109] ASSET, 2025, URL https://github.com/SWNRG/ASSET, (Accessed 6 March 2025).
[110] STEER, 2025, URL https://github.com/brunacordeiro/steer, (Accessed 6 March 2025).
[111] DENIS, 2025, URL https://github.com/SWNRG/DENIS-SDN, (Accessed 6 March 2025).
[112] SSDWSN, 2025, URL https://github.com/moalsaeedi/ssdwsn, (Accessed 6 March 2025).
[113] M. Kulkarni, M. Baddeley, I. Haque, Embedded vs. external controllers in software-defined IoT networks, in: 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), IEEE, 2021, pp. 298–302, http://dx.doi.org/10.1109/NetSoft51509.2021.9492688.
[114] Contiki OS, 2025, URL https://github.com/contiki-os/contiki, (Accessed 6 March 2025).
[115] Usdn-ng, 2025, URL https://github.com/mbaddeley/usdn-ng, (Accessed 6 March 2025).
[116] Contiki-ng, 2025, URL https://github.com/contiki-ng/contiki-ng, (Accessed 6 March 2025).
[117] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, N. Tsiftes, The contiki-NG open source operating system for next generation IoT devices, SoftwareX 18 (2022) 101089, http://dx.doi.org/10.1016/j.softx.2022.101089.
[118] E. Municio, J. Marquez-Barja, S. Latré, S. Vissicchio, Whisper: Programmable and flexible control on industrial IoT networks, Sensors 18 (11) (2018) 4048, http://dx.doi.org/10.3390/s18114048.
[119] E. Municio, S. Latre, J.M. Marquez-Barja, Extending network programmability to the things overlay using distributed industrial IoT protocols, IEEE Trans. Ind. Informatics 17 (1) (2020) 251–259, http://dx.doi.org/10.1109/TII.2020.2972613.

Liticia Djennadi earned her Master's degree in Computer Networks from Sorbonne Paris Nord University, France, in 2023. She is currently pursuing a Ph.D. at the same university, where she is affiliated with the L2TI Laboratory (Laboratory of Information Processing and Transmission). Her research focuses on next-generation networking technologies, particularly the Internet of Things (IoT), Software-Defined Networking (SDN), Network Function Virtualization (NFV), and routing protocols for Low-power and Lossy Networks (LLNs).



Gladys Diaz is associated professor, since 2001, at the University Sorbonne Paris Nord, France. She is member of the Network team at L2TI laboratory (Laboratory of Information Processing and Transmission). She received her Ph.D. degree in 2000 from INPL (Institut National Polytechnique de Lorraine), Nancy, France. She has obtained her HDR degree (Habilitation à Diriger les Recherches) in 2016 from the UPMC University (Université Pierre et Marie Curie, Paris 6), Paris, France. Her research interests are computer network and service architectures (SOA, Cloud Networking and Virtual networks), network management, QoS and information modeling, ITS, CloudFog and IoT architectures.

Khaled Boussetta is full professor at Université Sorbonne Paris Nord. He received a Ph.D. degree in Computer Science from University of Versailles in 2003 and an HDR (Habilitation à Diriger des Recherches) from Université Paris 13 in 2018. His is currently the head of L2TI laboratory. His research interests cover designing, modeling and performance evaluation of wireless networks. His recent research work focuses on IoT, vehicular networks, Edge/Fog/Cloud, infrastructures dimensioning and network mechanisms to support delay sensitive applications. He is co-author of 6 book chapters and 95 international publications. He is also actively involved in many top-tier conferences and journals.

Christophe Cérin is a Computer Science Professor at Sorbonne Paris Nord University since 2005, focusing on high-performance computing, including grid, cloud, and edge computing. He develops middleware, algorithms, and tools for distributed system management and explores computing's environmental impact. As an IEEE Member, he received the 2020 IEEE Technical Committee on Cloud Computing Research Innovation Award. He serves on the IEEE Transactions on Computers editorial board and as Associate Editor for Springer's Journal of Cloud Computing. Starting September 2021, he joined Grenoble University for research with the Edge Intelligence program and INRIA Datamove team.