Menu or a la carte? An architecture for programming the data plane of constrained wireless networks

Ahmad Mahmod, Julien Montavont, Thomas Noel

Menu ou à la carte ? Une architecture pour programmer le plan de donnée des réseaux sans fil contraints

Ahmad Mahmod[1] et Julien Montavont[1] et Thomas Noel[1]

[1]*ICube Laboratory (CNRS UMR 7357), University of Strasbourg, France*

Les réseaux sans fil contraints font face à un environnement dynamique, soumis à des variations constantes en raison de changements potentiels dans leur environnement physique, des évolutions des exigences applicatives, et des progrès technologiques. Ces variations peuvent affecter les performances du réseau et par conséquent nécessiter une reconfiguration de certains protocoles, voire un remplacement d'un protocole à la faveur d'un autre. Nous proposons dans cet article une architecture modulaire basée sur les réseaux programmables permettant de programmer de manière fine le plan de données des réseaux sans fil contraints. Une telle approche permet une adaptation constante du réseau aux variations dynamiques afin de maintenir le niveau de performance requis.

Mots-clefs : Low-Power Lossy Wireless Network, Software Defined Wireless Networks, Programmable Data Planes

1 Introduction

Low-Power Lossy Wireless Networks (LLWNs) are characterized by short-range, low-power, and low-datarate communications. They are particularly advantageous in various scenarios where traditional well-performing communication infrastructure (e.g. 5G networks) may be impractical or costly, such as environmental monitoring, industrial automation or healthcare systems. Those networks are dynamic due to the evolving nature of the wireless environment, energy constraints, and potential node mobility, all of which necessitate continuous adjustments to ensure efficient and reliable communication. Moreover, deploying alternative protocols becomes imperative when fundamental changes are needed to enhance performance or address new application requirements. This ensures that the communication infrastructure aligns with evolving demands and maintains optimal functionality. However, most protocols are hardcoded into the Operating System (OS), posing significant challenges when it comes to making protocol adjustments, such as modifying the radio configuration or replacing an existing protocol with a new one. While over-the-air firmware updates provide a solution for updating devices post-deployment, they often introduce interruptions due to the necessity of device reboots. Unfortunately, this rebooting process not only disrupts the continuity of operations but also contributes to high power consumption. This is primarily a result of the extensive exchange of messages required to bootstrap the network, involving tasks like neighbor discovery and route establishment.

This article introduces a novel architecture that makes the data plane of LLWNs fully *programmable*. Our architecture, leveraging Software Defined Networking (SDN), is inherently modular, providing the flexibility to modify individual protocol parameters (parametric programmability) or seamlessly replace one protocol with another (modular programmability). This adaptability ensures efficient customization and evolution of the network according to specific requirements. Our architecture is, to the best of our knowledge, the first to include radio management in addition to a packet processing pipeline. The contributions of this article are to i) review and compare the possible technologies for programming the data plane of LLWNs, and ii) propose a novel architecture to make the LLWN data plane programmable.

2 Existing Programmable Processing Pipelines

A programmable processing pipeline typically comprises three main stages — parsing, processing, and deparsing. The parsing stage is tasked with extracting and analyzing the packet header. The processing stage

processes the extracted headers and makes decisions according to the target of the processing. The deparsing stage recombines headers with the payload, preparing the new packet for the subsequent steps. These three stages are generally sufficient for wired networks, which benefit from a reliable and stable medium. However, in LLWN, MAC protocol plays a major role in the overall performance so, additional stages are required to manage the radio configuration for accessing the time-varying wireless medium. As a result, the inclusion of radio management in the framework is imperative for making the wireless data plane fully programmable. Multiple technologies emerged recently to provide a level of network programmability with different characteristics and use-cases such as Programming Protocol-independent Packet Processors (P4) [B+14], or extended Berkeley Packet Filter (eBPF) [V+20]. Other emerging technologies can be adapted to implement network programmability like Femto Containers (FCs) [Z+22].

2.1 Programming Protocol-independent Packet Processors (P4)

P4 is a high-level, domain-specific programming language designed for defining the data plane in network devices. P4 enables the processing of packets allowing for the definition of new protocols in a hardware-independent manner. This flexibility enables on-demand adaptive processing logic regardless of the networking device's underlying hardware. The architecture of P4 includes parsing the packets, making decisions for processing according to match-action tables (MATs), and finally, deparsing the packet to apply the processing output.

2.2 extended Berkeley Packet Filter (eBPF)

eBPF is a virtual machine (VM) that can be deployed in the Linux Kernel on-the-fly, enabling a wide range of applications beyond just packet processing, including monitoring, security, and analysis. The eBPF VM operates on an event-based model, utilizing *hooks* — specific checkpoints installed in operating systems to observe particular events. It is also lightweight, featuring small-footprint requirements ; 11 registers and a 512-byte stack, in addition to key-value maps for data sharing and storage. In terms of networking, eBPF defines a set of hooks covering different points of the network stack, starting from the driver level with the eXpress Data Path (XDP) hook to the higher layers with Traffic Control (TC) [V+20] enabling a tradeoff between performance and flexibility. However, the available actions for this low-level hook are very basic and are limited to dropping, redirecting, or passing to the stack.

2.3 Femto-Containers

Femto Containers (FCs) provide ultra-lightweight, secure, and isolated virtualization environments on IoT devices for various applications. They adapt the eBPF VM's concept and structure to Real-Time Operating Systems (RTOSs), making FCs lightweight and event-based. FCs are triggered using hooks in the OS but, unlike eBPF, with the capability to extend hooks to any checkpoint in the RTOS flow, from the driver level to the application level. However, the ultra-lightweight VM of FCs, with only 11 registers and a 512-byte stack in addition to a limited instructions set comparing to eBPF, restricts them to defining only basic functions within each container.

2.4 Comparison

LLWNs are constrained by limited memory, processing, and power capacities. Consequently, LLWNs require a lightweight technology with a small memory footprint and low power consumption. Additionally, this technology must not only provide a means to program the packet processing pipeline but also allow for complete control of the radio component, including functions such as switching on/off the radio and adjusting channel frequencies. Table 1 compares the proposed technologies for network programmability.

While P4 and eBPF present promising solutions for a programmable data plane, the absence of even minimal support for managing the radio chipset—a critical need in wireless networks—is attributed to P4's initial design for wired networks and eBPF's constrained interaction with device drivers, mainly through the basic XDP hook. Moreover, the hardware requirements for both P4 and eBPF do not align with the constraints of LLWN devices. P4 has demands on memory and processing capacities and eBPF is limited to Linux OS, which requires higher hardware capacities, which exceed what is typically available on LLWN devices. These factors contribute to the complexity of applying P4 and eBPF in LLWN environments.

TABLE 1: Comparison of Technologies

	P4	**eBPF**	**Femto-Container**
Scope	Domain-specific for data plane of network devices	Programming Linux Kernel including network stack	Event-driven applications in IoT devices
Footprint	Large memory and processing requirements	Small memory footprint	Small memory footprint
Limitations	Need high-perf. hardware, no radio management	Limited to Linux Kernel, no radio management	Limited performance for time-critical tasks

On the other hand, FCs offer a promising solution for deploying isolated network protocols in LLWNs, aligning well with the stringent requirements of network devices in such environments. FCs can install hooks for low-level system's events enabling a possibility to manage the radio. Also, FCs are lightweight with limited memory occupation in addition to reduce power consumption due to its event-based triggering nature. To create a full complex application, a chain of FCs can be utilized, with each container implementing an elementary part of the application. These applications, which could be communication protocols placed anywhere on the stack, are attached to the appropriate OS hooks. Importantly, these applications can be installed or updated at runtime without the need for firmware updates on the device. This characteristic offers a low-power and non-interruptive method to update the running services. Although FCs are hardware-agnostic but they are currently limited to specific RTOSs, and being a relatively new technology, they hold the opportunity for further development to become more OS-agnostic as well.

3 Proposed Architecture

LLWNs require a programmable data plane architecture that responds to the requirements of an adaptive wireless environment, while simultaneously respecting the constrained capabilities inherent to LLWNs. Our proposed architecture incorporates an *SDN Controller*, which represents a centralized control plane of the LLWN, leaving only the data plane in the LLWN devices. The SDN Controller continuously receives environmental conditions from the LLWN devices, analyzes these conditions, and then decides on the most suitable data plane protocols to perform optimally under such conditions and according to the required performance targets. The modifications may be limited to some parameters or may affect the whole deployed protocols. Implementing a centralized SDN architecture in LLWNs is particularly challenging due to unreliable links and network contentions that could impact control traffic. Moreover, ensuring successful updates and fast convergence requires transmitting modifications from the SDN controller to LLWN devices in a reliable and timely manner. This is crucial as all devices must promptly apply modifications to reestablish their communication capabilities. Allocating dedicated time-frequency blocks for the control traffic, as proposed in [V$^+$23], is one of the possible solutions to ensure a reliable control plane in LLWN.

The defined protocols of the data plane is deployed on the devices in the form of applications. To facilitate runtime modifications and ensure security, we propose defining these applications within lightweight Virtual Machines (VMs). Based on the previous comparisons, we nominate Femto Containers (FCs) to serve as the virtualization platform but, as we propose a general architecture, *any other **lightweight** virtualization technique may be used to achieve proposed architecture*.

FCs can be utilized to perform basic networking functionalities, leveraging the ability to install hooks on desired network events, including low-level ones. A micro-service approach can be adopted to create a packet processing pipeline, with each micro-service represented by an FC. For instance, to implement a simple link layer using the proposed approach, we may use a set of FCs to check the errors and parse the received frame header, then trigger another set of FCs that schedule the next hops, which precedes another set represents the deparser that forwards the received frame to the next hop (see Fig.1 (a)). This use-case proves that the FCs micro-services approach can implement the processing pipeline supported by P4 and eBPF. Furthermore, our approach enables pre-processing management of the wireless radio device using the proper *timing* hooks. Fig.1(b) demonstrates a direct duty-cycle management of the wireless device use case, which cannot be implemented using P4 and eBPF.

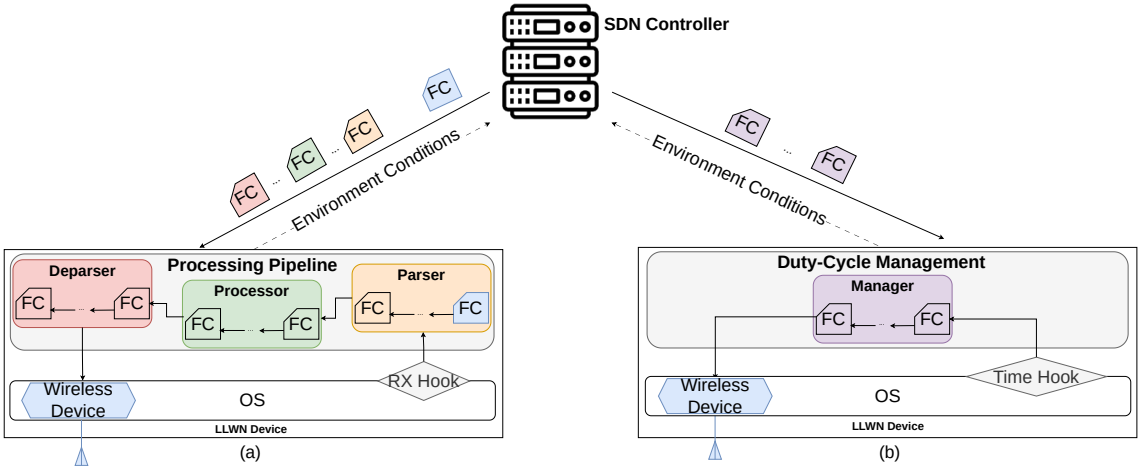

FIGURE 1: Proposed Architecture

Our architecture is adaptive and double-modular, enhancing scalability, facilitating easy updates, and enabling fault isolation. The first level of modularity resides within the protocol itself, where each elementary function in an FC (micro-service) operates independently of the others. This means that each function can be updated without impacting the others. Fig.1(a) shows how the Parser is updated by replacing a single FC without affecting the others. The second level of modularity exists between the protocols, which are independent chains. This allows for the protocols to be updated separately. For example, the Deparser may be completely replaced with a new one. However, our system may experience performance degradation compared to monolithic and hardcoded implementation of network protocols, especially when deploying synchronous protocols that demand precise timings. This will be explored further in future research.

4 Conclusion and Future Works

In this work, we have highlighted the need for network programmability in LLWNs due to their dynamic environments. We reviewed and compared some network programming technologies and after this reviews, we proposed a **double-modular** and **lightweight** programmable network architecture for LLWNs. We found that FCs may offer a promising solution for our programmable architecture in LLWNs by providing the programmable pipeline in addition to the **radio management**. Our architecture is based on SDN and utilizes lightweight VMs in a micro-service approach to define the data plane in LLWN devices. We are currently implementing the processing pipeline using FCs on real hardware as a proof-of-concept then we are planning to implement the radio management to provide a testbed for the proposed architecture.

Acknowledgments

This work is part of the ANR-funded project PERENNE (ANR-23-CE25-0008).

Références

[B+14] P. Bosshart et al. P4 : programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3), 2014.

[V+20] M. Vieira et al. Fast Packet Processing with eBPF and XDP : Concepts, Code, Challenges, and Applications. *ACM Computing Surveys*, 53(1), 2020.

[V+23] F. Veisi et al. Enabling Centralized Scheduling Using Software Defined Networking in Industrial Wireless Sensor Networks. *IEEE Internet of Things Journal*, 10(23), 2023.

[Z+22] K. Zandberg et al. Femto-containers : lightweight virtualization and fault isolation for small software functions on low-power IoT microcontrollers. In *proc. of the ACM/IFIP International Middleware Conference*, 2022.