Towards an FPGA-Based Network Layer Filter for the Internet of Things Edge Devices

T. Gomes, F. Salgado, S. Pinto, J. Cabral and A. Tavares
Centro Algoritmi - University of Minho
{tiago.m.gomes, filipe.salgado, sandro.pinto, jorge.cabral, adriano.tavares}@algoritmi.uminho.pt

***Abstract*—In the near future, billions of new smart devices will connect the big network of the Internet of Things, playing an important key role in our daily life. Allowing IPv6 on the low-power resource constrained devices will lead research to focus on novel approaches that aim to improve the efficiency, security and performance of the 6LoWPAN adaptation layer. This work in progress paper proposes a hardware-based Network Packet Filtering (NPF) and an IPv6 Link-local address calculator which is able to filter the received IPv6 packets, offering nearly 18% overhead reduction. The goal is to obtain a System-on-Chip implementation that can be deployed in future IEEE 802.15.4 radio modules.**

***Index Terms*—Internet-of-Things (IoT), Packet Filter, System-on-Chip, FPGA, 6LoWPAN, IPv6, Contiki-OS.**

I. Introduction

Everyday new smart devices are getting connected to the internet, building the so-called Internet of Things (IoT). Daily usage objects are becoming smarter and start to play a key role in our everyday life [1]. From a complex Smart City Monitoring System to a simple Smart Street Lamp or from an advanced security system to a Patient Vital Signs Monitoring System [2,3], all these “Things” have a common basic requirement: connectivity. A device which cannot communicate and interact with other devices is often considered as limited and soon will be seen as useless. However, the exponential growth of the IoT infrastructure leads to several challenges, among these: scalability and interoperability [4]. Scalable and standard communication protocols will better fulfil these requirements [5,6], as standard protocols target the interoperability, contributing for a rapid development by easily enabling heterogeneous devices to communicate.

The internet, as we know it, cannot address such big number of connected devices. In spite of the Internet Protocol (IPv4) providing a good infrastructure and robust protocol to reach devices from anywhere, it cannot provide unique global reachability as it is limited to 32-bit of singular addressable interfaces and it was not initially designed to handle such kind of devices. IPv6 is the key for connecting myriads of smart devices on the new internet era [7]. Initially conceived to support scalability, with 128-bit for unique addressing along with other enhanced and new features, allows to all devices to be singly identified and reachable any time from anywhere.

The Low Rate Wireless Personal Area Networks (LR-WPANs), whose forming nodes are predominantly resource constrained (in terms of memory, processing capabilities, power, etc.) are mainly IEEE 802.15.4 based networks. The IEEE 802.15.4 standard has been widely adopted by well-known technologies like ZigBee and, up till now, proved to be the best for implementing the physical (PHY) and MAC layers. All devices should be able to participate in the complex IoT network. Thus, in order to use the IPv6 standard with the current technology, the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [8] was specially developed allowing this protocol to be used over the IEEE 802.15.4 data frames, implementing all the packet features such as fragmentation and reassembly, packet header compression, Internet Control Message Protocol (ICMP), etc. Per contra, enabling all devices to be reachable through an unique IPv6 address over a 6LoWPAN network has a drawback, as the overhead caused by the adaptation layer may affect the performance of the small devices. When the full stack is implemented (Network Layer with routing capabilities and the Transport TCP/UDP Layer), formerly not used by the IEEE 802.15.4 based networks, the performance of these devices will drastically decrease.

Alike some basic features of the MAC layer being already implemented by some modern radio transceivers (freeing some processing overhead from the device’s CPU), the 6LoWPAN protocol features can also be offloaded to hardware. These radios are able to implement basic IEEE 802.15.4 network processing, like filtering some packet header fields, e.g., the Personal Area Network (PAN) address. This simple feature contributes to a more efficient CPU working time, as the processing of unwanted MAC data frames can be discarded.

Some approaches for offloading Network Stack capabilities have already been attempted, e.g., in [9] an “RTOS in hardware for energy efficient Software-based TCP/IP Processing” is proposed while in [10] specific packet processors have been used to implement specific applications with specific needs. However, concerning the Network Layer and for the best of the authors knowledge, this work in progress paper goes beyond the state-of-the art, presenting an FPGA-based solution for the Network Layer of the IoT networks. This Work-in-Progress (WiP) focuses on the connectivity of the IoT Edge Devices (EDs) by implementing features of the Network Layer in dedicated hardware such as the packet filtering and processing functionalities, aiming to increase the performance and thus improve the overall efficiency. This approach aims to evaluate, as a proof-of-concept, future efficient SoC implementations with the new improvements and added features.

978-1-5090-1314-2/16/$31.00 ©2016 IEEE

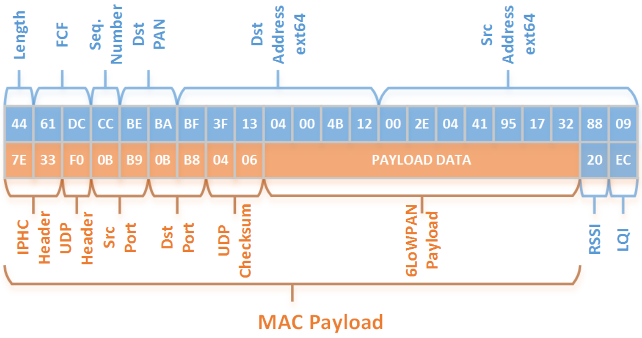

Figure 1. IEEE 802.15.4 Data frame example

II. Network Packet Filter

This work presents a hardware Network Packet Filter (NPF) accelerator which is able to implement important features such as the filtering and processing of the incoming IP packets. Fig. 1 depicts an example of an IEEE 802.15.4 Data frame that carries the IPv6 packet. For the IP address filtering, the NPF is able to filter, by accepting or discarding, the incoming packets regarding the destination and/or source IP address. Due to the used IPv6 compression (obtained on the IPHC header field [11]), the MAC Data frame header is needed to generate the IP addresses from the *Dest Addr* and *Source Addr* fields. For instance, based on the IPHC header information value given in Fig. 1 (0x7e33), the IPv6 addresses in use are fully compressed and must be calculated as follows: the 128-bit IPv6 address is computed by filling the first 64-bit with the Link-Local address prefix followed by zeros, and the last 64-bit must be obtained from IEEE 802.15.4 MAC address. So the computed Link-Local IPv6 address is: fe80::02:12:4b:00:04:13:3f:bf. The calculation of the address is significantly faster in hardware than in software due to the number of needed iterations, thus this feature is also implemented by the NPF accelerator.

Following the same approach for the Transport Layer on the same example, the UDP header can be also processed and the *Source Port* and *Dest Port* can also be used to discard or accept incoming packets by processing these fields. This simple approach can, at several levels of the Network stack, promptly pre-process and filter the received packets, adding another level of security from the point of view of the connectivity and availability of the ED. That means, for instance, when unwanted packets (intentionally or for a different destination) are received, if the destination or source IP address matches the configured addresses, the filter will still drop a packet if no UDP connection is listening on the received ports.

One can easily understand the consequence of the unwanted packets received by the ED, which may permit intentional attacks to a target ED by increasing a system overload causing a well known Denial-of-Service (DoS) attack [12]. Under such condition, the normal services provided by an ED will drastically impact performance or even stop running.

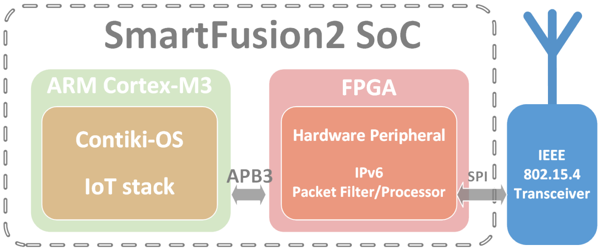

Figure 2. System architecture

The mentioned features are proposed to be implemented in two ways, as the incoming packets may be of two different types: (1) packets to be accepted or (2) packets to be discarded by the *Source Addr* or the *Dest Addr* field. This allows to implement the NPF with a novel concept of packet filtering, by adding IP addresses to a White List Address (WLA) or to a Black List Address (BLA) and filtering them accordingly, which means addresses on the WLA will be accepted while those ones on the BLA will be discarded. However, only one approach can be used at a time.

If a device plays the role of a Network Router (NR), the NPF gains a role of higher importance over the network. Usually the traffic is higher on a NR, overloading the available bandwidth in periods of high traffic. Adding these routing capabilities to the NPF, the performance and availability of the EDs would considerably increase, allowing more efficient operating modes and resources saving. For the best of the authors knowledge, this novel features were not yet proposed nor implemented in hardware.

III. Implementation

In order to implement the proposed system (Fig. 2), the Microsemi's SmartFusion2 Security Evaluation Kit was selected. This platform consists of a cost effective SoC FPGA which integrates flash-based FPGA fabric and a 166 MHz ARM Cortex-M3 processor, along with many other features.

The hardcore processor runs the Contiki-OS [13], an IoT open-source Operating System (OS). It provides a full Network Stack implementation with all the protocol standards and layers, offering a variety of software applications and examples, contributing for a better software development. The Contiki-OS interacts with the hardware peripheral using the Advanced Microcontroller Bus Architecture (AMBA) 3 Advanced Peripheral Bus (APB) protocol and the peripheral connects with the selected IEEE 802.15.4 radio transceiver (CC2520) using a Serial Protocol Interface (SPI) bus.

A. NPF hardware accelerator

The NPF accelerator implements all the aforementioned features described in section II. When configured and enabled, the peripheral is able to detect when the radio transceiver holds a valid packet on the RX FIFO, to transfer the packet to an internal memory and to process it according to the configured fields. If the transferred packet is accepted and validated, the NPF interrupts the OS execution to notify the reception of a

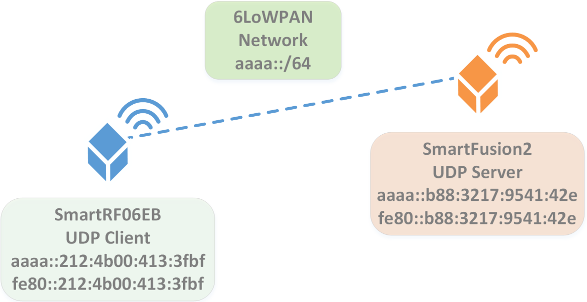

Figure 3. Test Scenario

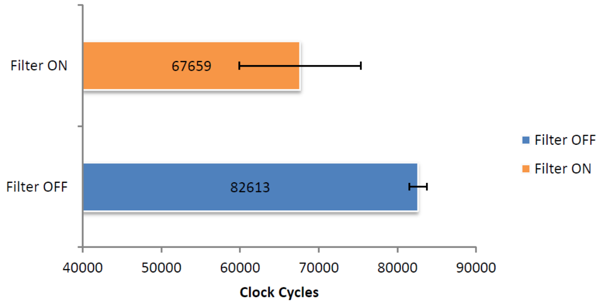

Figure 4. Performance evaluation

new IP packet, otherwise the NPF drops the packet and the OS continues its normal execution. If the notification is ignored by the OS, the NPF will run normally, giving priority to newly arrived packets and overriding the previous ones, ignored by the OS. When the OS requests a packet transfer from the NPF, the module will give priority to the OS request and if a new packet arrives, it will be held in the radio RX FIFO until the current transfer ends.

The NPF module also implements registers that can collect and hold statistics about the filtered/dropped packets according to the selected fields. These registers can be read any time by the OS, providing extra information about the module status.

B. *Contiki-OS*

Despite of Contiki-OS supporting several hardware platforms and architectures, there is no support for the selected hardware platform, therefore a software porting was made in order to run the proposed features. All the platform dependent OS modules (timers library, peripherals, etc), were created and added to the OS in the form of software libraries, providing a new platform support to the active Contiki community and allowing code re-use by future developments. A peripheral driver was built and integrated with the OS radio driver. This OS agnostic implementation allows an easy integration with other OSes that implement the IoT Network Stack, e.g., the RIOT-OS [14].

At the current stage of research two features are fully implemented by the NPF: (1) IP source address filtering and (2) IP destination address filtering.

IV. Preliminary Results

In order to evaluate the implemented features, the testing scenario depicted by Fig. 3 was developed. Concerning the packet exchange between the two nodes, an UDP Server runs on the SmarFusion2 platform and an UDP Client is implemented by a SmartRF06EB platform with a CC2538EM. To perform tests and comparing results, the UDP Server runs with the NPF turned ON (filtering by destination IP address) and OFF (software will handle the packet discard). When the NFP is ON the UDP Client IP address is added to the WLA and it will be accepted by the NPF and delivered to the OS network stack. Other addresses than the UDP Client IP address will fail the WLA and will be dropped by the NPF, keeping the OS free for other running processes, i.e., the OS will not be interrupted when the NPF notifies the reception of new packets.

A. *Performance Evaluation*

The conducted test consists of a simple benchmark where the delay (in clock cycles) caused by the action of filtering the incoming IP packets is measured, both in the software and hardware solutions. To evaluate the NPF performance, the UDP Client is configured to send valid Over-The-Air (OTA) packets to the UDP Server at a fixed rate of 32 packets/s.

The results from this test are illustrated in Fig. 4. With the NPF OFF the number of clock cycles needed to filter one IP packet is, on average, 82613 and with the hardware module ON the average is reduced to 67659, which represents an overhead reduction of 18,1% (1.22× speed-up). The standard deviation (2δ) is, respectively, 7729 and 1134. The results suggest that just by offloading this feature to hardware, the ED can accelerate the processing of valid received IP packets. For the dropped packets, the OS will take the same time to discard the packet. However if the NPF is ON, the time to take this action is not consumed by the OS, leading to the test in the section IV-B, where the impact of non-valid packets on the OS execution performance is evaluated.

B. *System's Availability Evaluation*

This test consists of running Thread-Metric Benchmark Suite [15] in order to evaluate how the NPF alleviates the OS overhead. This benchmark suite measures the time taken by RTOSes to perform specific services, e.g., cooperative and preemptive context switching and (preemptive) interrupt processing. Although Contiki supporting preemptive and cooperative modes, the preemptive mode is not yet supported in the ARM Cortex-M3 architecture. Thus, only the cooperative context switching test was conducted. The benchmark creates five OS processes and one report process which periodically prints the benchmark value. With NPF turned ON and OFF on the UDP Server, the UDP Client sends packets to the UDP Server and changes the *Dest Addr* field (to be filtered by the NPF) at different packet sending rates (4, 8, 16, 32, 64, 128 and 256 packets/s).

Fig. 5 illustrates the results from the Thread-Metric benchmark test with the NPF OFF. The benchmark value decreases

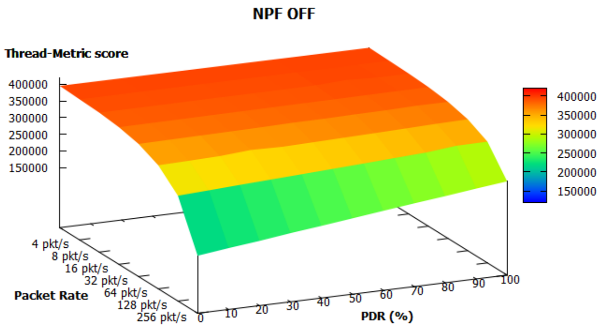

Figure 5. Systems availability with NPF OFF

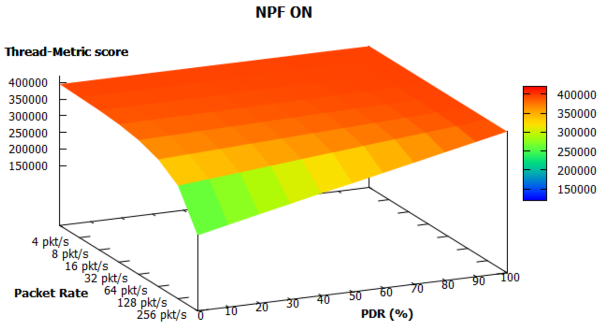

Figure 6. Systems availability with NPF ON

as the the packet sending rate from the UDP Client increases. This is due to the overhead caused by the OS tasks responsible to receive and process the received packets, at all the stack layers. Increasing the Packet Discard Rate (PDR), that is, the number of packets to be filtered by the NPF, the benchmark tends to perform better but still, the filtering and discarding tasks, at the MAC and Network Layers have to be performed. The second test, with the NPF ON (Fig. 6), show higher benchmark results as the PDR increases. This represents the overhead reduction caused by the NPF on discarding the incoming packets whose *Dest Addr* do not match the WLA. With a PDR of 100%, the benchmark result is the same as with a packet rate of 0 packets/s. This represents the highest system availability because the filtered packets by the NPF are not received by the OS, thus the associated OS tasks are not scheduled to run.

V. Research Roadmap

Further research will focus on implementing all the proposed features into the NPF, aiming to provide, as a proof-of-concept a set of features and functionalities to be used by future IEEE 802.15.4 radio systems for the IoT devices.

Other layers will be analysed and more candidates will be selected to be implemented on the NPF. For the Network Layer, the routing algorithm and the routing tables are seen as good candidates, accelerating the routing process and increasing the OS performance. Also, the OS availability may be increased as the incoming packets to be forwarded to a different destination, will be processed by the NPF instead the OS.

VI. Conclusions

The number of devices connected and sharing data over the IoT network will highly increase in the near future. Aiming to increase the EDs performance and contribute to a more efficient OS execution, this WiP proposes a novel solution to deal with three of the main challenges for the new IoT devices: (1) connectivity, (2) scalability and (3) interoperability.

The performed tests of the proposed NPF have shown a overhead reduction of 18,1% as well an improvement on the OS availability on processing other tasks. In our opinion, the NPF offers a good solution to be deployed in future IEEE 802.15.4 radio transceivers as the 6LoWPAN is accepted as the new standard to provide IPv6 connection to the constrained EDs.

VII. Acknowledgements

Tiago Gomes is supported by FCT - Fundação para a Ciência e Tecnologia (grant SFRH/BD/90162/2012). This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope: PEst-UID/CEC/00319/2013.

References

[1] L. Mainetti, L. Patrono, and A. Vilei, “Evolution of wireless sensor networks towards the internet of things: A survey,” in *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, Sept 2011, pp. 1–6.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[3] L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and L. Tarricone, “An iot-aware architecture for smart healthcare systems,” *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515–526, Dec 2015.

[4] M. B. et all, “Deliverable D1.5 - Final architectural reference model for the IoT v3.0,” Internet of Things - Architecture, Tech. Rep., 01 2015.

[5] C. Cees Links, “White paper: Wireless Communication Standards for the Internet of Things,” GreenPeak Technologies, Tech. Rep., 01 2015.

[6] A. Foster, “White paper: Messaging Technologies for the Industrial Internet and the Internet of Things,” PrismTech, Tech. Rep., 01 2015.

[7] I. Cisco Systems, “White paper: Integrating an Industrial Wireless Sensor Network with Your Plant’s Switched Ethernet and IP Network,” Cisco Systems, Inc, Tech. Rep., 01 2009.

[8] J. Olsson, “6LoWPAN demystified,” Texas Instruments, Tech. Rep., 10 2014.

[9] N. Maruyama, T. Ishihara, and H. Yasuura, “An rtos in hardware for energy efficient software-based tcp/ip processing,” in *Application Specific Processors (SASP), 2010 IEEE 8th Symposium on*, June 2010, pp. 58–63.

[10] F. Hijaz, B. Kahne, P. Wilson, and O. Khan, “Efficient parallel packet processing using a shared memory many-core processor with hardware support to accelerate communication,” in *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on*, Aug 2015, pp. 122–129.

[11] J. Hui and P. Thubert, “Compression format for ipv6 datagrams over ieee 802.15.4-based networks,” Internet Requests for Comments, RFC Editor, RFC 6282, September 2011, http://www.rfc-editor.org/rfc/rfc6282.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6282.txt

[12] A. D. Wood and J. A. Stankovic, “Denial of service in sensor networks,” *Computer*, vol. 35, no. 10, pp. 54–62, Oct 2002.

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

[14] RIOT. The friendly Operating System for the Internet of Things. [Online]. Available: https://www.riot-os.org/

[15] I. Express Logic. Thread-Metric Benchmark Suite. [Online]. Available: http://rtos.com/downloads/articles_and_white_papers-1/