Far East Journal of Electronics and Communications
© 2017 Pushpa Publishing House, Allahabad, India
http://www.pphmj.com
http://dx.doi.org/10.17654/EC017040877
Volume 17, Number 4, 2017, Pages 877-887 **ISSN: 0973-7006**

RIOT-OS: FIRMWARE FOR FUTURISTIC INTERNET OF THINGS

Illa Ul Rasool, Yousaf Bin Zikria, Arslan Musaddiq, Farhan Amin and Sung Won Kim*

Department of Information and Communication Engineering
Yeungnam University
South Korea
e-mail: illaulrasool@ynu.ac.kr
yousafbinzikria@ynu.ac.kr
arslan@ynu.ac.kr
farhanamin10@hotmail.com
swon@yu.ac.kr

Abstract

Internet of things redefine the modern networking by interconnecting meaningless things in a more substantial way. It works towards paving a path between things and technology. However, as things can scale from small to relatively large, there exist certain constraints. These constraints are classified as low memory footprint, low power CPU, etc. Therefore, limiting the constraint factors of these devices is a major challenge. An efficient operating system, however, can serve the purpose of IoTs in future. Subsequently, RIOT-operating system can operate with limited IoT hardware resources available and can be considered as future de facto IoT-OS. Our work represents the RIOT-

Received: March 13, 2017; Accepted: May 2, 2017
Keywords and phrases: RIOT, NDN, CS, FIB, PIT.
A preliminary version of this paper was presented at the ICIDB-2016, Seoul, South Korea.
*Corresponding author
Communicated by Gyanendra Prasad Joshi

OS study with respect to more recent network technology, named data networking. The study mainly considers the resource constraints of IoT devices and RIOT-OS features-implementation to mitigate them. Further, a combined architecture of NDN-RIOT with smart tick less timer scheduling mechanism to increase energy efficiency of devices is discussed. Other relevant features of RIOT supporting existing transport protocols are also studied.

1. Introduction

The use of technology and devices has reformed the way humans work. It is the modern phase of technology where electronic devices can no more afford to stay isolated and disconnected from the world. This motivation is driving the devices or things towards interconnectivity forming a network of things or internet of things (IoT) [1]. IoT is a network of things which ranges between real-time devices such as wireless sensors, medical devices, security cameras, and even large infrastructures and vehicles. The significance of IoT powered devices is that they can be accessed remotely from anywhere, as well as command other devices to perform necessary tasks. Although these devices increase the physical world, awareness among devices is constrained in: (i) available memory, (ii) processing power, (iii) communication and (iv) battery life time [2]. With all constrains considered, these devices are nevertheless required to adapt to the physical world and perform necessary real-time processing.

Table 1. RIOT features [3]

Features	RIOT-OS
Minimum RAM	✓
Minimum ROM	✓
C support	✓
C++ support	✓
Multi-threading	✓
MCU without memory management unit	✓
Modularity	✓
Real-time	✓

Acknowledging all the requirements of IoT devices and the related constrains, an IoT-operating system (OS) must be capable enough to withstand such incompatible and varying hardware properties. RIOT-OS is a solution to IoT devices spanning across low power to energy efficient microcontroller units. RIOT-OS exhibits full support for IoT devices in terms of minimum RAM, ROM requirement [4]. It also provides developer friendly environment for C and C++ programmers. RIOT-OS micro-kernel supports the core mechanisms such as multi-threading, priority-based scheduling interrupt handling, and inter-process communication. It is accurately synchronized with hardware to provide real-time execution of applications. For a summarized information, Table 1 provides a brief overview of RIOT-OS features. With all the features and requirements for IoT devices in check, this paper focuses on RIOT-OS in terms of named data networking (NDN) scheme [3]. Also, referred as content centric network (CCN), NDN is a data name based networking scheme. The motivation behind NDN is to achieve synchronism between Internet architecture and its usage. It means that not all Internet services are sufficiently supported by the current built in architecture. The idea of NDN resides within the ambitious information-centric network (ICN) project which follows the content/information/data centric approach [3]. The overall goal of data/content/information oriented network is to redefine the current host-centric Internet architecture (TCP/IP) into information-centric. It is clear enough that IoTs are expected to be implemented extensively in future. Also, the corresponding OS and wireless network are going to play an important role in its development and application process. NDN-RIOT for IoT devices itself define a total overhaul of present wireless technology into more convenient and less complex one. The design goal of NDN-RIOT is: (i) to support the basic NDN-forwarding mechanism, (ii) to support the current protocol regulations, (iii) NDN-RIOT-OS must suit memory constrains of IoTs, and (iv) full support for CPU that runs at a clock speed less than 100MHz [4]. The contribution of our work introduces the RIOT-OS for IoTs, and presents the architecture for RIOT-OS. Moreover, it analyzes the NDN-RIOT architecture-mechanism and features with an NDN application. This paper discusses in Section 2 the features of RIOT-OS for IoTs. Section 3 describes the basic mechanism of NDN. Section 4 features NDN-RIOT architecture and example run of NDN in RIOT. Finally, Section 5 concludes the paper.

2. RIOT: Architecture

Earlier, the OS for wireless sensors and for Internet hosts differed with respect to available memory, energy efficiency, modularity of kernel, and API access. RIOT-OS goal is to fulfill the varying requirements of IoTs that require overall OS reliability and availability of corresponding C and C++ libraries for developers. RIOT offers the real multi-threading processing inherited from FireKernel [4]. Thus, FireKernel contributes to the real-time and modularity requirements of IoTs. It features zero-latency interrupt handlers, and minimum context-switching times along with threading priorities.

Moreover, every microcontroller unit has a particular scheduler which wakes up the system at certain time instants. Schedulers refer to the timers for such waking purpose. This process is called the *timer tick*. However, there can be certain instants when devices are not required to perform tasks and they are triggered by scheduler to wake up. To avoid this, RIOT introduces a tickless scheduler which allows the device to switch to idle state and enter deep-sleep mode thus increasing the energy efficiency [5]. Adding to this, the devices can be interrupted in deep-sleep mode by external interrupts or kernels only. Further, efficient energy output of an OS is dependent on the degree of complexity of kernel processes. Further, kernel functions depend upon the duration and occurrence of context-switching. RIOT-OS mainly performs the thread switching by an interrupt. Therefore, it is important to reduce the amount of thread switch-time-under-process. For that instant, RIOT introduces minimized scheduler, which after finishing the interrupt service routine does not require saving old thread-context. This process significantly reduces task-switching processing time.

3. Named Data Network

The NDN communication follows a specified pattern or a path to which the data plays the catalyst role rather than concentrating on host-client identities (IP-based). In NDN, the content or data is defined by a specific name which regulates its recognition and retrieval by the destination application. The NDN architecture constitutes of a consumer and producer. The consumer sends a data-request in the form of an interest packet. Interest packet has a data name as its prefix. The data packet is forwarded in the network following the interest name and interest forwarding strategy. The forwarding strategy is particularly based on the forwarding information base (FIB). Every forwarding node has an FIB to make the calculative decisions about the potential destinations from where data can be retrieved. The data-destinations are actually the nodes that provide the requesting consumers with requested data. The node that satisfies the consumer interest is referred as provider. Meanwhile, during the forwarding process of interest packet, each node in the routing path keeps the track of the interface from which the interest arrived in its pending interest table (PIT). Also, each forwarder maintains a temporary data cache called *content store* (*CS*). CS is used to cache the data earlier sent to satisfy a specific interest. Furthermore, an interest can find its requested data either at the real provider or in the CS of any forwarder. Finally, when the interest is satisfied, the data follows the same routing path as interest packet earlier recorded in the PIT of every forwarder. Further, the data packet constitutes of a cryptographic signature that allows the consumer to authenticate the content. The main emphasis is not on the provider or node but the content or data produced by the provider. Figure 1 represents the basic NDN networking scheme and forwarding strategy.

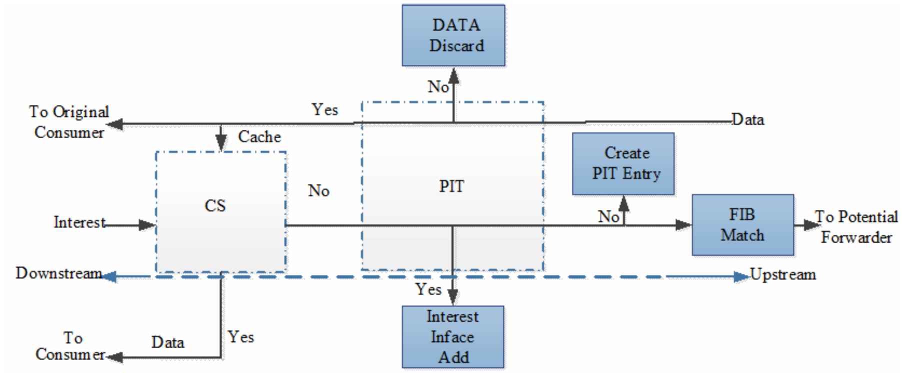

Figure 1. Named data network forwarding scheme.

4. NDN-RIOT

(a) Aim

Aim of NDN-RIOT-OS is to fully support the named packet forwarding procedure of NDN, as well as it must obey the present protocol regulations. NDN is a futuristic technology and most probably will be implemented in IoTs.

NDN-RIOT supports such devices with 10s of Kb of RAM (executional data store), 100s of Kb of flash memory, and low power central processing unit running at a clock speed of less than 100MHz. With such memory constraints, the NDN-RIOT-OS must execute and support all core mechanisms and functionalities of NDN such as FIB, PIT and CS.

(b) Architecture

RIOT-OS implements micro-kernel architecture as kernel threads. Figure 2 shows that these threads represent network protocols: IPv6, UDP. Inter-process communication (IPC) is responsible for the communication between kernel threads. NDN-RIOT constitutes of threads: application, NDN, network device driver [4]. When an application requires to send a particular data, it forwards the packet to NDN layer via IPC. Further the packet is processed and transferred to network device driver for transmission.

In case of packet reception, it follows the reverse path till packet reaches application layer. The above process occurs in the presence of an interrupt handler, and other peripherals. RIOT-OS handles the memory constraints in NDN-devices by storing the NDN packets in wire format throughout.

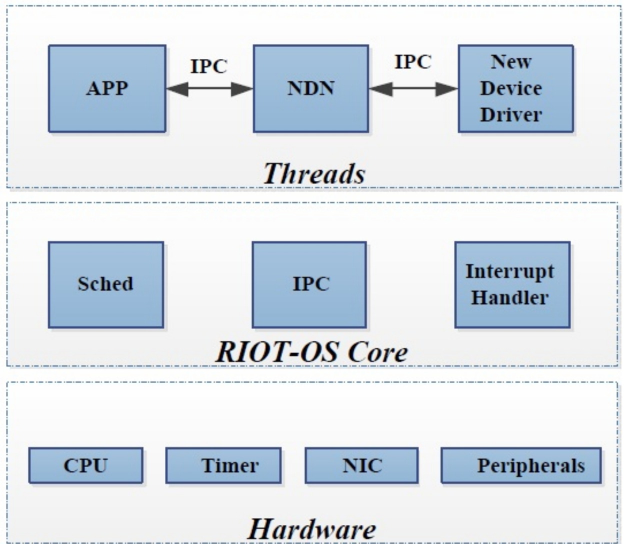

Figure 2. NDN-architecture and inter-process communication between threads.

(c) Packet forwarding

The NDN communication follows a specified pattern or path to which the data plays the catalyst role rather than concentrating on host-client identities (IP-based). In NDN, the content or data is defined by a specific name which regulates its recognition and retrieval by the destination application. In NDN, every forwarding node has an FIB to make the calculative decisions about the potential destinations from where data can be retrieved. The data-destinations are actually the nodes that provide the requesting consumers with requested data. The node that satisfies the consumer interest is referred as provider. Meanwhile, during the forwarding process of interest packet, each node in the routing path keeps the track of the interface from which the interest arrived in its pending interest table (PIT).

Furthermore, each forwarder maintains a temporary data cache called *content store* (*CS*). CS is used to cache the data earlier sent to satisfy a specific interest. Therefore, an interest can find its requested data either at the real provider or in the CS of any forwarder. Finally, when the interest is satisfied, the data follows the same routing path as interest packet earlier recorded in the PIT of every forwarder. Further, the data packet constitutes of a cryptographic signature that allows the consumer authenticates the content. The main emphasis is not on the provider or node but the content or data produced by the provider. Figure 1 represents the basic NDN networking scheme and forwarding strategy. In basic NDN-forwarding strategy, an interest packet when received by any neighboring node performs the following process. Firstly, it checks the availability of data in its own CS following a lookup process. In case the data is found, it is dissipated back to the consumer through same interface the interest arrived earlier. However, in case of no data found, node performs the PIT lookup. During PIT lookup process if a corresponding entry is found for the same data, then it discards the interest and saves the incoming interface. Nevertheless in case of non-PIT matchup, a PIT entry is created for the interest and is further forwarded to the FIB. Consequently, FIB preforms calculative based decision to search for a potential data-provider. Secondly, when interest finds a data match at a certain provider, a data packet under same interest-name-prefix match is transverse back. The data packet again follows the same interface/s on the way back to consumer. Moreover, on the way back, each forwarder will authenticate the data by checking the corresponding PIT entry earlier stored. Further, when data reaches the original consumer, it runs a signature validation test to confirm/authenticate the data received.

Example run

We run a simulatory example for NDN-RIOT scenario, where we create five nodes to simulate basic NNDN working in RIOT. Figure 3 shows that how tap interface feature of Linux allows the NDN-ping application to see a raw network traffic. We assign tap 0 to consumer and tap 3 to producer nodes, respectively. Figure 4 shows that how consumer expresses an interest in an NDN named format. Figure 5 shows the packet acknowledgement by producer and replies back with data after a named prefix match. On the reception of data by the consumer, it checks the data authentication after a signature validation process. In this manner, RIOT is able to express the core functionality of NDN network by basing communication on data name rather host-centric (IP-based).

```
user@instant-contiki:~/riot/RIOT/dist/tools/tapsetup$ ./tapsetup --create 5
creating tapbr0
creating tap0
creating tap1
creating tap2
creating tap3
creating tap4
```

Figure 3. Node creation for scenario.

```
user@instant-contiki:~/riot/ndn-riot-examples/ndn-consumer$ ./bin/native/ndn_con
sumer.elf tap0
RIOT native interrupts/signals initialized.
LED_RED_OFF
LED_GREEN_ON
RIOT native board initialized.
RIOT native hardware initialization complete.

main(): This is RIOT! (Version: 2016.07-devel-443-g49d46-instant-contiki)
client (pid=2): enter 's' to start
s
client (pid=2): express interest, name=/ndn/p%F8%02%E5
client (pid=2): enter app run loop
client (pid=2): data received, name=/ndn/p%F8%02%E5/%03
client (pid=2): RTT=662us
client (pid=2): content length = 22
client (pid=2): signature valid
```

Figure 4. Consumer interest/data expressing/receiving.

Figure 5. Producer acknowledging interest and sending data packet.

5. Conclusion

The basic architecture of RIOT-OS for IoT devices is briefly discussed. RIOT-OS provides memory, and energy efficient features as well as modularity and real-time support for IoT devices. RIOT is also programmable and provides developer with C and C++ libraries. Further, RIOT-OS is studied in accordance with NDN, which is a futuristic wireless network technology. NDN-RIOT provides an overhaul of concept of how “internet of things” can work without any memory and energy constraints in future. Further, the RIOT is compatible with the existing protocols (IPv6, RPL) only contributes for future convenience. The architecture for FIB, CS and PIT is a basic one implemented in RIOT due to memory constraints. However, comparatively NDN works efficiently with respect to RIOT and future work can include implementing full data names.

Acknowledgment

This research was supported in part by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-R2718-16-0035) supervised by the IITP (National IT Industry Promotion Agency) and in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01058751).

References

[1] L. Atzori, A. Iera and G. Morabito, The internet of things: a survey, Computer Networks 54(15) (2010), 2787-2805.

[2] A. Sehgal, V. Perelman, S. Kuryla and J. Schonwalder, Management of resource constrained devices in the internet of things, IEEE Communications Magazine 50(12) (2012), 144-149.

[3] W. Shang, A. Afanasyev and L. Zhang, The design and implementation of the NDN protocol stack for RIOT-OS, NDN, Technical Report, NDN-0043, July 16, 2016.

[4] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch and T. C. Schmidt, RIOT-OS: towards an OS for the internet of things, 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Apr. 14, 2013, IEEE, pp. 79-80.

[5] K. Roussel, Y. Q. Song and O. Zendra, RIOT-OS paves the way for implementation of high-performance MAC protocols, Proceedings of the 4th International Conference on Sensor Networks, ESEO, France, 2015.