Function Oriented Network Architecture for Realization of Autonomic Networks

Gijeong Kim, Sungwon Lee*
Department of Computer Engineering
Kyung Hee University
Youngin-si, Korea
{kimgijeong, drsungwon}@khu.ac.kr

***Abstract*—Existing networks have focused on high data transfer rate and reliable data delivery through lower header processing and effective signaling. Recently, reducing CAPEX/OPEX (Capital Expenditures/Operating Expenditure), deploying new network services, and orchestration and management are issues of growing importance in the network. To solve these issues, we propose a Function Oriented Network (FON) that can facilitate the rapid deployment of new network functions and protocols to satisfy the requirements that are generated continuously over time. In this work, we introduce the FON architecture and describe the implementation of the FON prototype. Through this implementation, we demonstrate flexible network programmability and support core architecture to realize autonomic networking.**

Keywords-Autonomic Networks; Bio-inspired Networking; Network Programmability.

I. INTRODUCTION

Existing networks, such as the Internet, have focused on high data transfer rate and reliable data delivery. To achieve these goals, the network should perform lower header processing and should have effectively designed signaling. Over time, advances in hardware and transmission technology have allowed the user to support a high quality of data transmission.

Recently, reducing CAPEX/OPEX, deploying new network services, and orchestration and management are issues of growing importance in the network. In order to solve such problems, there are typical network paradigms such as Software Defined Networking (SDN), Network Function Virtualization (NFV), and autonomic networking [1][2][3].

To reduce OPEX/CAPEX and realize an autonomic network, we designed and implemented the Function Oriented Network (FON) that realizes a biology inspired autonomic network and can flexibly deploy new network services. Moreover, it enables the user and the service provider, as well as the network operator, to control the user plane of the network. The FON targets large scale Internet-of-Things (IoT) and access networks.

The rest of the paper is structured as follows. In Section II, we introduce the FON architecture and in Section III we describe its implementation. Section IV presents the conclusion and future work.

II. FUNCTION ORIENTED NETWORK ARCHITECTURE

In this study, we propose the FON architecture and SmartPackets to achieve network programmability by facilitating the rapid adoption of new network functions in network devices by network operators, service providers, and users.

In FON, a network node provides functions for network services on the basis of software and functions for network services, which are executed in terminals, where they are specified using open functions and transferred via SmartPackets. In addition, network services are managed by updating, distributing and removing functions for network services, which are executed in network devices via the network manager. Figure 1 provides an overview of the FON architecture and the structure of a SmartPacket. The FON comprises the FON device, FON node, and FON manager.

A. FON Device

The FON device is a database (DB) that stores function tables with function names and function codes, and a function processor, which creates SmartPackets and transfers them to a FON node, as well as receives SmartPackets from a FON node. The function processor receives function distribution messages from the FON manager, and stores and updates the function table information in the FON manager, including the message in the function table of the FON device. The function processor can perform SmartPacket verification using the function tables stored in the DB during SmartPacket generation. Through this procedure, a FON device can request and execute network functions in the FON node.

B. FON Node

A FON node is a DB that stores function tables, which contain executable function names, function code, and function usage counts (statistics). A function processor receives a SmartPacket and extracts information from the function call field to call the function that corresponds to the extracted function name, thereby performing the function. The function processor extracts the function code that needs to be executed from the DB and it performs dynamic binding to execute dynamic binding functions. In addition, the function processor can specify the function execution result in the SmartPacket payload and it transfers SmartPackets to other FON devices or FON nodes.

 ISBN: 978-1-61208-450-3

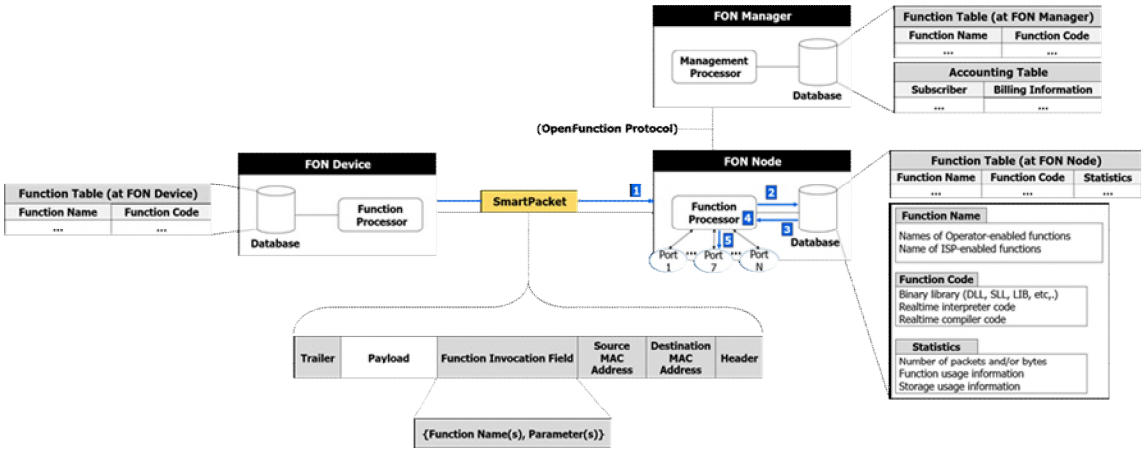

Figure 1. Function Oriented Network Architecture and SmartPacket Structure

C. *FON Manager*

The FON manager has a DB, which contains a function table with executable function names and function code, and an accounting table that manages billing information according to the function usage of the users and their subscriber information. The FON manager also has a management processor that extracts the function table from the DB and transfers it to the FON device or a FON node.

The management processor adds new functions that can be executed in a FON node to the function table of the FON manager and it transfers the function addition message, thereby adding the function name information and function code information to the function table of the FON node. The management processor transfers a function removal message to the FON node, thereby removing a specific function name information and function code information from the function table stored in the FON node. The management processor receives function usage information for each user from the FON node and it calculates billing information based on the function usage per user. It stores and manages this information in the accounting table.

D. *SmartPacket*

As shown in Figure 1, a SmartPacket is composed of the destination MAC header, the source MAC header, the function invocation field, the payload, and the trailer. The function invocation field includes the function name information that needs to be called, along with the required input parameters. It may contain multiple function names and input parameters. The output parameters for the execution results of functions that need to be called can be inserted into the payload.

III. IMPLEMENTATION

To implement the FON architecture, we built the testbed using a virtual environment, such as Virtual Box, and we implemented the FON Device and the FON Node using Linux virtual machine. The FON Manager was not implemented as part of this work and will be implemented in the future. The resources allocated for the virtual machine were Xeon E5520 2.27GHz CPU, 812MB RAM, 32GB SSD, and Gigabit Ethernet NICs. The FON Node and the FON Device were developed using the Python 2.7.9 programming language. The SmartPacket can be involved in many network function calls. Therefore, it is necessary to perform encoding and decoding of variable length message. The SmartPacket structure was implemented by using the pickle library of Python. Basically, FON should be implemented as an upper layer of MAC, but our prototype was implemented using TCP/IP protocol, such as socket programming. Ultimately, it will be implemented to L2/L3 based protocol.

IV. CONCLUSION AND FUTURE WORKS

In this paper, we introduced the FON architecture and described its implementation. The FON architecture can facilitate the rapid deployment of new network functions and protocols to satisfy requirements that are generated continuously. In future works, we will carry out performance evaluation and analysis through simulation and verification experiments. These experiments will target the processing overhead and message overhead, and will compare our approach with similar active network architectures [4]. After that, we will design and implement an ant-colony optimization based autonomic network using FON.

ACKNOWLEDGMENT

This work was supported by the ICT R&D program of MSIP/IITP, Republic of Korea. (B0101-15-1366, Development of Core Technology for Autonomous Network Control and Management), (B0190-15-2013, Development of Access Technology Agnostic Next-Generation Networking Technology for Wired-Wireless Converged Networks)

REFERENCES

[1] Naudts, Bram, et al. “Techno-economic analysis of software defined networking as architecture for the virtualization of a mobile network,” 2012 European Workshop on Software Defined Networking (EWSDN), Oct. 25, 2012, pp. 67-72.

[2] Hawilo, Hassan, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal “NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC),” IEEE Network, volume 28, issue 6, Nov. 24, 2014, pp. 18-26.

[3] M. Behringer, B. Carpenter, et al. “A Reference Model for Autonomic Networking,” draft-behringer-anima-reference-model-04, IETF, Oct. 16, 2015, pp. 1-24.

[4] David L. Tennenhouse and David J. Wetherall, “Towards an Active Network Architecture,” ACM SIGCOMM Computer Communication Review, volume 26, issue 2, Apr., 1996, pp. 5-17.

 ISBN: 978-1-61208-450-3