# *A Plug-in Node Architecture for Dynamic Traffic Control*

Wangbong Lee, Dong Won Kang, Joon Kyung Lee
Division of Next Generation Internet Research
Electronics and Telecommunications Research Institute
Daejeon, Korea
emails {leewb, dwkang, leejk}@etri.re.kr

***Abstract*—Accurate traffic classification is fundamental to many network activities such as an analysis of the application usage, anomaly detection, and accounting. However, the architecture of most network devices is strictly fixed. We need a dynamic architecture to deal with increasing traffic. We therefore propose a plug-in node for dynamic traffic control. A plug-in node is designed to load and unload traffic processing plug-in modules on the fly. Plug-ins can be implemented with various functions including packet classification, anomaly detection, and application monitoring.**

***Keywords-Plug-in; Traffic Classification; Performance model.***

## I. INTRODUCTION

A greater understanding of Internet traffic has become more important over the past few years [1]. Moreover, mobile data traffic is expanding more rapidly now than ever before. The number of mobile subscribers and applications continues to increase globally [2]. To deal with the current expansion of data traffic, network/mobile operators need to optimize their networks and collect key performance data through traffic measurement and monitoring techniques.

Research on Internet traffic classification has generated creative and novel approaches. Accurate traffic classification is fundamental to many network activities [6] such as an analysis of application usage, anomaly detection, and accounting. A correct analysis of Internet applications can provide valuable information to network operators for building efficient networks. Detecting anomalies in a network is a critical activity for network operators and end users in terms of service availability. Traffic classification is also important for application-based billing and user-based services.

A work by Arthur Callado et al. introduced and classified traffic classification techniques [1]. Summarizing this research, there is no silver bullet technique in terms of accuracy and completeness. A work by Alberto Dainotti et al. discusses recent achievements and future directions in traffic classification [3]. This research suggests several strategies for tackling unsolved challenges. Their recommended strategies focus on a mixture of traffic classification techniques, speed increments of network links, rigorous evaluations, and open source implementations.

In 2011, KaKaoTalk, one of most famous social network service (SNS) application in Korea, overloaded the 3G mobile network through its highly frequent control messages [4]. One major Korean network operator, SKT, recognized the importance of network traffic optimization, and they proposed a network traffic mitigation method [5]. However, newly proposed techniques are not easy to apply to current network devices such as access points, edge routers, and gateways

For this research, we designed a traffic classification system supporting high-speed links based on a commercial network processor and its architecture. The system has a dynamic plug-in node for supporting various traffic control functions.

Internet traffic is always changing. The variety and complexity of modern Internet traffic exceeds anything imagined by the original designers of the fundamental Internet architecture. As the Internet has become our most critical communications infrastructure, service providers are attempting to improve the functionality, including security, reliability, privacy, and multiple service qualities, into a best-effort architecture. To prioritize, protect, or prevent certain traffic, service providers need to implement a technology for traffic classification. In this paper, a plug-in system is proposed for adopting various traffic classification techniques. This system is designed for a user-defined network service using a network processor based network interface card (NIC). Our proposed conceptual model of a plug-in node is shown in Figure 1.

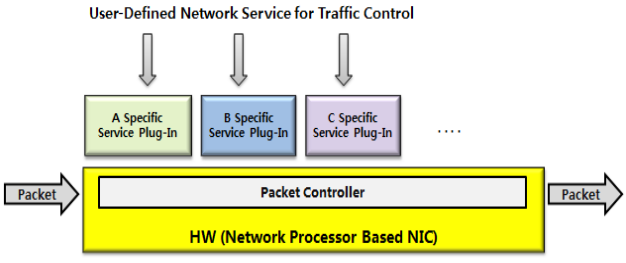



Figure 1. Conceptual model for a plug-in node

The packet controller is similar to a network switch. It must perform packet classification at high speeds to efficiently implement various functions. Packet classification requires matching each packet against predefined rules, and forwarding the packet according to the matching results. When a packet is matched against predefined rules, it is forwarded to a specific plug-in module that is dynamically loaded or unloaded by users. Service plug-in modules

 ISBN: 978-1-61208-239-4 

include a Peer to Peer (P2P) traffic control plug-in, traffic classification plug-in, traffic monitoring plug-in, and so on. Service plug-ins can be implemented as traffic classification techniques. Thus, the plug-in node is considered a multi-traffic classifier system. A newly proposed method of traffic identification can apply to proposed node owing to its plug-in feature.

The rest of this paper is organized as follows. We begin by reviewing proposed plug-in node architecture in Section II. In the next section, we provide the test result in its prototyping system. Finally, some concluding remarks and a description of future work are given in Section IV.

## II. PLUG-IN NODE ARCHITECURE

To deal with the rapid growth of Internet traffic, current backbone network devices such as routers and switches have to manipulate millions of packets per second at each port. To achieve such a high packet processing rate, hardware devices such as an application-specific integrated circuit (ASIC), field-programmable gate array (FPGA), and ternary content-addressable memory (TCAM) are usually adopted in the design of current network devices. Current Internet traffic processing systems need to support a variety of emerging network applications while guaranteeing a high packet processing rate. To achieve the requirement of a high packet processing rate, network processors are introduced as a promising solution for building network devices [7]. Our proposed node is implemented on a Tilera network processor board in a general x86 Linux server. The Tilera network processor board is designed as an NIC, and has a peripheral component interconnect express (PCI-e) interface and 4 x 10 Gb Ethernet ports, as shown in Figure 2 [8].

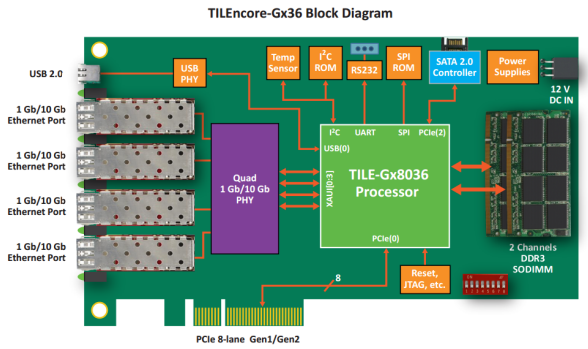



Figure 2. Tilera network interface card

We designed the main components, which manipulate packets and hardware resources, into a network interface card. The service plug-in adapter manages the plug-ins and isolates their resource space, which prevents abnormal crashes from creating a failure in another plug-in. Figure 3 shows the main components: a command line interface (CLI) parses the user's commands and transmits the control messages to the appropriate modules. A virtual switch implements the general switch functions including queue management and packet forwarding among the interfaces ports. It manages the packet flows through a user-defined rule, which is based on the 6-tuple classification rule. The plug-in manager starts and stops the plug-in modules using their service profilers. The resource monitor reports the hardware resource information such as memory usage and processor usage. The connector is a communication channel between the service plug-in adapter (SPIA) and service plug-in panel (SPIP), which is implemented in x86 Linux server, as shown in Figure 4. This channel uses a PCI-e interface and external management Ethernet interface.

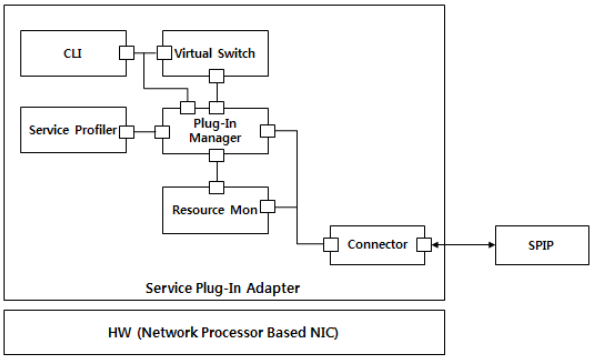



Figure 3. Service plug-in adapter

A service plug-in panel consists of a platform manager, service manager, and log manager. The service manager controls the plug-ins using their service profile information. Users can build plug-in programs and their own profiles using the software developer's kit (SDK) [8]. Plug-in SDK uses a Tilera network processor's compiler and simulator, as well as plug-in libraries. The log manager reports and saves errors and resource information from the service plug-in adapter. The connector has the same functionality as the service plug-in adapter.

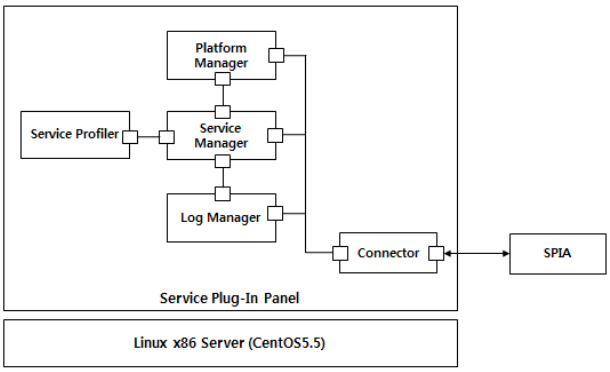



Figure 4. Service plug-in panel

## III. EVALUATION

We implement the plug-in node to demonstrate its feasibility. This proof of concept is implemented on a Tilera multi-core processor. We implement the core functions of the plug-in node including the switch function, plug-in

 ISBN: 978-1-61208-239-4 

manager, and resource manager. Figure 5 shows the test topology. Various types of traffic are generated by tcp replay running on a regular Linux operating system [9][10]. The results of the experiments are shown in figure 6. The average latency of a 1,024-byte packet for five plug-ins is 9.7 msec, while that of a 1,024-byte packet for no plug-in is 2.4 msec.

Chertov et al. [11] estimated that a perfect router has under a 0.2 msec packet delay for a 1,024-byte packet, while a Cisco router has under a 0.35 msec packet delay. A perfect router is hypothetical, as it has zero processing and queueing times. Our current experiment values are not adequate to apply the commercial product. The current prototype uses Linux network APIs for a proof of concept. Thus, we have room to optimize the average latency during the development phase

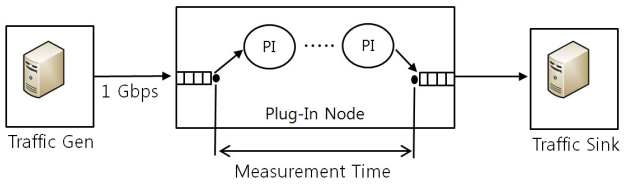
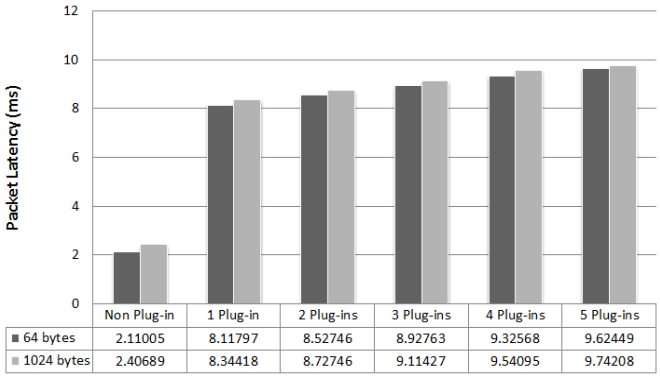



Figure 5. Test Topology




| | Non Plug-in | 1 Plug-in | 2 Plug-ins | 3 Plug-ins | 4 Plug-ins | 5 Plug-ins |
|---|---|---|---|---|---|---|
| 64 bytes | 2.11005 | 8.11797 | 8.52746 | 8.92763 | 9.32568 | 9.62449 |
| 1024 bytes | 2.40689 | 8.34418 | 8.72746 | 9.11427 | 9.54095 | 9.74208 |

Figure 6. Packet Latency in Plug-in node

## IV. CONCULSION AND FUTURE WORK

The goal of this study was to provide a plug-in node architecture. It is designed to load and unload traffic processing plug-in modules during runtime. Plug-ins can be implemented with various functions including packet classification, anomaly detection, and application monitoring. In this paper, we implemented the prototype of plug-in node and provided the test result. Our current experiment values are not adequate to apply the commercial traffic control product. We are going to optimize the code using network processor specific APIs during the development phase.

We plan to evaluate the performance of our proposed node under various realistic application workloads using Tmix [10]. Also, we are going to study on a packet latency analysis model based on a queuing system to evaluate our proposed node.

For our future work, we consider the software defined networking (SDN) in terms of dynamic traffic control. Thus, we are going to research on how to interact SDN control plane such as NOX, and how to implement SDN packet forwarder as a plug-in.

## REFERENCES

[1] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, “A Survey on Internet Traffic Identification,” Communications Surveys & Tutorials, IEEE , vol. 11, no. 3, pp. 37-52, 3rd Quarter 2009

[2] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html [retrieved: July, 2012]

[3] A. Dainotti, A. Pescape, and K. C. Claffy, “Issues and future directions in traffic classification,” *Network, IEEE* , vol. 26, no. 1, pp. 35-40, January-February 2012

[4] http://www.seoul.co.kr/news/newsView.php?id=20110402015019 [retrieved: July, 2012]

[5] M. W. Kim, D. G. Yun, J. M. Lee, Y. J. Shim, and S. G. Choi, “Network traffic mitigation method using TCP signalling delay algorithm,” Advanced Communication Technology (ICACT), 2012 14th International Conference on , vol., no., pp.730-733, 19-22 Feb. 2012

[6] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” Proceedings of the 2005 ACM SIGMETRICS, pp. 50-60, June 06-10, 2005, Banff, Alberta, Canada

[7] http://en.wikipedia.org/wiki/Network_processor [retrieved: July, 2012]

[8] http://www.tilera.com/ [retrieved: July, 2012]

[9] http://tcpreplay.synfin.net/ [retrieved: July, 2012]

[10] M. C. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, and F. D. Smith. “Tmix: A tool for generating realistic application workloads inns-2,” ACM Computer Communication Review, vol. 36, pp. 65–76, July 2006.

[11] R. Chertov, S. Fahmy, and N. B. Shroff, “A Device-Independent Router Model,” *INFOCOM 2008. The 27th Conference on Computer Communications, IEEE* , pp. 1642-1650, 13-18 April 2008

 ISBN: 978-1-61208-239-4 