Grainflow: A Per-bit Customizable Scheme for Data Plane Innovation on Programmable hardware

Zhongjin Liu, Yong Li, Li Su, Depeng Jin, Lieguang Zeng
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
jindp@mail.tsinghua.edu.cn

ABSTRACT

In this paper, we introduce Grainflow, a scheme for flexibly deploying custom protocols at wire-speed. Grainflow is designed based on programmable hardware along with configuring software. Grainflow allows operation on every bit of header to be customized. With different configuration, Grainflow can flexibly customize IP and Non-IP protocols on the fly.

1. INTRODUCTION

Next generation network has been intensively researched and various of novel protocols have been proposed. These protocols cover different layers and require modification on data-plane of routers and switches. Openflow[2] provides researchers with a chance to deploy protocols based on 10-tuples. However, some non-IP protocols' packets don't conform to this pattern. Thus heterogenous protocols are impossible to deploy on Openflow. Switchblade[1] integrates common packet-processing modules, enabling different protocols to be deployed using common functions. However, common functions are limited to all the modules predefined by developer. Switchblade has to resynthesize hardware if new functions are about to be added. We present Grainflow, a programmable hardware platform enables operation on per header bit to be customized, which allows dynamically modifying functions without having to resynthesize hardware.

2. ARCHITECTURE

The architecture of Grainflow is depicted in Fig. 1. There are two kinds of flows in Grainflow, one is data flow used for packet processing and the other is control flow for software to access registers. The hardware

ACM CoNEXT Student Workshop, December 6, 2011, Tokyo, Japan.

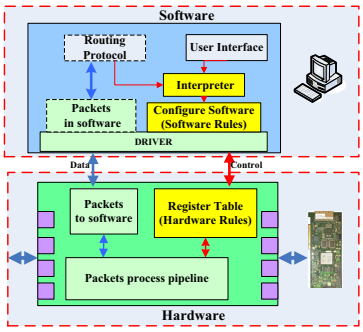

Figure 1: Architecture of Grainflow

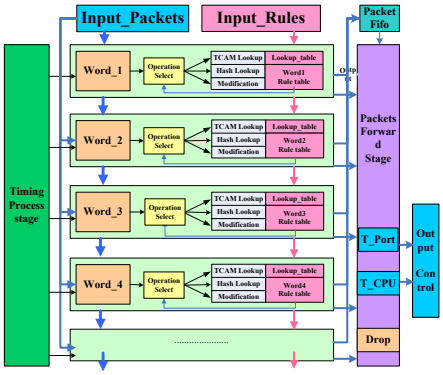

Figure 2: Grainflow Pipeline

plays a key role in processing packets, and it holds a pipeline and a set of register tables that control all the modules inside the pipeline. Software mainly contains four parts: drivers for hardware; routing software which runs routing protocol; user interface that allows users to configure user-define rules and an interpreter that translates routing tables and the user-define rules into hardware register addresses and values. To decouple data and control flows, We build hardware in FPGA and software in common PC, which are connected by PCI bus.

2.1 Grainflow pipeline

The data-plane of Grainflow is designed as a pipeline architecture shown in Fig. 2. A timing process module outputs timing signals that indicate which word of input

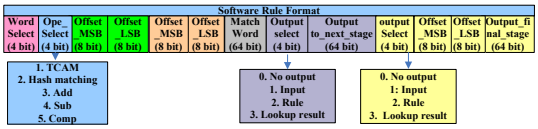

Figure 3: Customizing Rule

packet is appearing at the 64-bit bus. We divide the Grainflow pipeline into various stages according to these signals.

We exploit data processing procedure and find out a set of common modules involving all the operations to the packet header. These modules consist of the following operations. TCAM searches for results among mass of entries. Hash matching is used for comparing certain packet field with predefined value. Modification can be done to some packet fields, like adding or subing on certain bits. At present, two kinds of tables are employed: A lookup table that is designed for the TCAM module, since the TCAM module lookups tables conforming to a unique pattern; A rule table that stores rules translated from user-defined rules. Under the control of rules, each stage can realize specific function. The output data of every stage has two directions to go, one is to next stage and the other is to final forwarding stage. When a stage is activated by the timing module's signal, it first checks all the items in the rule table to select operations to process the word. To ensure the precise simultaneous output, Grainflow introduces the parallel item handling. Every common module except TCAM has several duplications to process different items.

2.2 Customizing Rules

The customizing rules are shown in Fig. 3. To provide full flexibility in configuring the hardware, a single rule has up to twelve fields. Every field is defined as follows: (1) Word select: the first 4 bits indicate which stage should this rule work at; (2) Operation select: 4 bits are used for selecting operations. Grainflow employs the options for TCAM, hash matching, add, sub and compare; (3) First offset: 8 bits offset pair indicates the rule's cover area of word from upper stage; (4) Second offset: This pair indicates which part of incoming packets data remains to be processed; (5) Match word: This value is used for hash matching, and operating number for add or sub; (6) First output select: This value is for Selecting output pattern to next stage, and options are no output to next stage, output the input packet word to next stage, rule defined value and lookup result; (7) Output sent to next stage: This word will be output to next stage if last field choose 2; (8) Second output select: Selecting output pattern for final forwarding stage; (9) The last offset define this rule taking effect for which output bits; (10) Output sent to final stage: This word's value will be output to final stage if field 8 chooses 2.

IP_Router_Rule												
Wd _sel	Fun _sel	1st Offset		2nd Offset		Match Word	Output to next stage		Output to final stage			
1	2	0	0	63	15	MAC_ADDR1	0	0	2	10	0	00100000001
1	2	0	0	63	15	.....	0	0	2	10	0	.....
1	2	0	0	63	15	MAC_ADDR4	0	0	2	10	0	00101000000
1	2	0	0	63	15	0xFFFFFF	0	0	2	10	0	1100000000
2	2	0	0	63	48	0x0806	0	0	2	1	0	01
2	2	0	0	63	48	0x0806	0	0	2	1	0	10
3	5	0	0	47	40	1	0	0	2	0	0	0
3	4	0	0	47	40	1	0	0	3	8	1	0
4	0	0	0	15	0	0	1	0	0	0	0	0
5	1	15	0	63	48	0	3	0	3	40	0	0
6	1	31	0	0	0	0	0	0	3	31	0	0

Figure 4: Configuration scripts for IPv4

3. CUSTOMIZE AN IPV4 ROUTER

To demonstrate the flexibility of deploying protocols, we instance a data-plane of IPv4 router . Our prototype system is implemented in NetFPGA, which has four ethernet ports, and each has a corresponding MAC address.

The configuration scripts are shown in Fig. 4. The toal number of rules is twelve without considering of checksum. For the first stage, there are five rules to parse header. These rules are used for indicating if the packet is broadcast style and packet's initial output port. At the second stage, *ethertype* at the packet header is checked to see whether the packet is ARP or IP packet. The third stage's two rules make sure that TTL value is larger than 1 and decreases the value by 1. The fourth stage directly output the last 16 bits of input to next stage. These bits combine with the first 16 packet bits of stage 5 to make up a destination IP address which is used for longest prefix matching. The result of next hop IP is exported to next stage, which complete the function of ARP lookup. The finals stage gather all the result to judge whether the packets should be dropped or forwarded. By this means, we can use other simple configuration scripts to different protocols.

4. SUMMARY AND FUTURE WORK

We have presented the design and the main idea of Grainflow and shown this novel architecture's flexibility for deploying protocols. Grainflow provides researchers with high flexibility to deploy their own IP or non-IP protocols without resynthesize hardware, which saves development time. However, its scalability remains to be researched, which remains as our future work.

5. REFERENCES

[1] M. B. Anwer, M. Motiwala, M. B. Tariq, and N. Feamster. Switchblade: A platform for rapid deployment of network protocols on programmable hardware. In *ACM SIGCOMM*, Aug 2010.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in college networks. 2008.