Newly Structured Router Network Architecture using Cloud Control Plane and Forwarding Plane

Sho Shimizu, Shan Gao, Daisuke Ishii, Naoaki Yamanaka
Department of Information and Computer Science, Keio University,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan,
Email: shimizu@yamanaka.ics.keio.ac.jp

Abstract—This paper proposes a novel network architecture called cloud control plane (Cloud C-plane) to reduce the power consumption of networks. In Cloud C-plane, the forwarding functions and control functions are decoupled, and the control functions are placed in the cloud. In addition, physical network topology is modified to improve the energy efficiency when the load of traffic changes. The routing engine on a dynamically reconfigurable processor (DRP) is a key component of Cloud C-plane. We implement the prototype of the routing engine on an actual DRP, and the experimental results show the execution time of the prototype is 19 times faster in the shortest path calculation.

I. INTRODUCTION

The power consumption of networks is one of key issues in the future network architecture. In fact, the power consumption of network equipments is rapidly increasing. The estimates for the U.S. power consumption by the Internet is a range between 2 and 8 percent [1]. Significant growth in this power consumption is expected as many devices connect to the Internet. As a result, the power consumption of network equipments will be equal to one nuclear reactor of a power plant. A new network architecture is strongly required to drastically reduce the power consumption of networks. Under such situation, some researches to improve energy efficiency in the current network architecture have been studied recently. [1]–[15].

In this paper, we propose a new network architecture, where the forwarding and control functions of a router are decoupled. Decoupling makes networks more advanced since the speed of evolution of their technologies are different. It is better to decouple the control function from the router itself. When the control and forwarding functions are decoupled, we can aggressively optimize these functions independently, especially control functions. Separating forwarding and control functions is suitable for cloud computing. Control functions can be placed in the cloud. We call this architecture cloud control plane (Cloud C-plane). In the network with Cloud C-plane, control functions can be running on a powerful server. Thus, sophisticated traffic engineering can be executed.

The physical topology of forwarding plane varies when the load of the network changes in order to reduce the power consumption of the network. Advertising the physical topology change does not occur to prevent the network from being unstable. The advertised network topology is independent from the physical topology, and the physical topology is optimized in terms of energy efficiency.

Path calculation is a key in Cloud C-plane, then we employ routing engine on Dynamically Reconfigurable Processor (DRP) there. This approach is based on an on-chip emulated network. The reason that we use DRP in the routing engines is to implement experiment based traffic engineering algorithm, which is a new approach to calculate a suitable path. One of the merits of DRP is flexibility. The internal circuit structure of DRP can be designed, then we make an emulated network, which corresponds to the real network, on DRP. Emulated packets are transmitted throughout the emulated network, and we can experimentally get the shortest path because the first emulated packet from the source node and the destination indicates the shortest path. We develop the prototype of the routing engine on an actual DRP.

This paper as organized as follows. Section II describes the proposed architecture of Cloud C-plane. In Section III, we explain the basic algorithm of the proposed routing engine on DRP. The prototype implementation of the routing engine is shown in Section IV, and the experimental results are provided in Section V. Finally, we summarize this paper in VI.

II. CLOUD CONTROL PLANE

A current router consists of two main functions; forwarding functions and control functions. In this paper, we call a system where the forwarding functions are running a forwarding element, and a system where the control functions are running a control element. Currently, a control element corresponds to the operating system of a router, such as IOS and JUNOS. A forwarding element and a control element are tightly coupled in a current router as shown in the left figure of Fig. 1. On the other hand, in our proposed network architecture, they are decoupled as shown in the right figure of Fig. 1. The speed of evolution of a forwarding element and a control element is different. Thus, it is better that they make progress independently between each other. If control and forwarding functions are decoupled, control elements don’t care about the architecture of forwarding elements, then we can easily replace forwarding elements with more energy efficient forwarding elements if these are available. In addition, aggressive network optimization can be applicable in the decoupled architecture. It is the reason why we employ decoupling.

---

[1]–[15]: These references are cited in the paper to support the claims made in the text. They provide further details and context for the research presented. The specific content of these references is not visible in the provided text.
Forwarding and control functions are tightly coupled

Forwarding and control functions are decoupled

A control element have the corresponding forwarding element. The control element is responsible for tasks to control and manage the forwarding element such as routing. A control element communicates with the forwarding element by the protocol called Forwarding Element Control Protocol (FECP), which is similar to GSMP [16] or OpenFlow Protocol [17]. Decoupling control and forwarding functions have been also considered in the IETF ForCES (Forwarding and Control Element Separation). It is an interface between forwarding elements and control elements. For example, a control element sets the forwarding configuration via FECP. Control elements make a control plane, and forwarding plane make a forwarding plane.

The location of a control element is independent from that of the corresponding forwarding element. Figure 2 shows that the control elements are placed in the cloud. Control elements can be virtualized as software service because they are decoupled with the corresponding forwarding elements. We call this architecture cloud control plane (Cloud C-plane). As a result, control elements are running on virtual machines, and they are placed in a server in a data center.

In the Cloud C-plane architecture, a physical network topology, which means a real network topology of forwarding elements, is changeable to optimize the total power consumption of networks. However, the advertised network topology, which is a topology advertised by a routing protocol such as OSPF, does not change due to maintain stability of networks. The advertised network remain unchanged even when the physical topology changes. Physical Topology Controller (PTC) is responsible for decisions about turning on or off forwarding elements. Each control element monitors the amount of traffic through its corresponding forwarding element, and periodically reports the traffic information to PTC. Consequently, PTC knows the load of all links in the network, and all information about the physical topology. PTC determines which forwarding elements have to be powered off or on according to the current network state. In Fig. 2, the forwarding element F2 is going to be powered off, then, the functions of F2 is virtually swapped to F3 to maintain the advertised topology. PTC configures the control elements to complete the above procedures without topology modification on the advertised network.

Figure 3 shows the physical topology modification when the amount of traffic changes. The number of powered-on routers and links is large when the network load is high. On the other hand, the number of powered-off routers and links increases when the network load is low.

PTC and control elements placed in the cloud are important components of our proposal. As shown in Fig. 2, each control element is running on the virtual machine, and several virtual machines are placed in a server. To speed up path computation in these control elements and optimization in PTC, we propose that the server equips a routing engine. The engine can be used among the control elements on virtual machines. We propose that the routing engine is based on Dynamically Reconfigurable Processor (DRP). The detail architecture of the proposed routing engine is described in the next section.

III. ROUTING ENGINE ON DRP

Recently, technology of reconfigurable devices, such as Field Programmable Gate Array (FPGA) and Dynamically Reconfigurable Processor (DRP), have been developed [18]. We can design a dedicated hardware for own purpose by using these types of devices. They are very attractive due to achieving high performance originating from hardware implementation. In addition, DRPs have flexibility that we can dynamically change a internal circuit to another circuit at high speed, for example, within a few clock cycles. We
take advantage of dynamic reconfigurability of DRPs for the routing engine.

Our proposed approach is to make an on-chip emulated network, which corresponds to the real network. We transmit emulated packets through the emulated network and we observe the behavior of links or routers on the emulated network. We can experimentally get the optimization result from this observation. Figure 4 shows the real network and the corresponding emulated network on DRP. The real network consists of links and routers, and then we make emulated links and routers on the emulated network.

Two types of emulated packet are defined. The first type of the emulated packet is called emulated flooding packet (EFP). An EFP has three main fields: the first field is to record the index number of the source router, the second field is to record the bandwidth of a traffic demand from the source router, and the third field is to record the index number of a link. Every emulated router can get global information from EFPs sent on the emulated network. The second type of the emulated packet is called emulated path search packet (EPSP). An EPSP has two main fields: bandwidth recording field and link index number recording field. The former field is used to record the bandwidth of a passed link. The latter field is used to record the index numbers of all passing links.

Three metrics, such as delay, link utilization and bandwidth, are considered in our TE method and the method operated on an emulated network on DRP. For considering the link utilization, the link cost is changed dynamically according to the number of passed packets. For example, packet count function is added in emulated links to count the number of passed packets. When the number of passed packets surpass a threshold set by a network administrator, the link cost will be increased. Therefore, this link will not be chosen in the next time of searching the shortest path. The bandwidth is a parameter of an emulated link and both EFPs and EPSPs have a field to record the bandwidth of a link. Each emulated link generates several clock cycles of delay according the real link cost.

Figure 5 shows an example of parameter initialization which uses EFPs. In this example, two EFPs pass through the emulated link #2, where the third field is abbreviated. One EFP is from the emulated node X and uses 80 Mbps as a bandwidth. Another EFP is from the emulated node Y and uses 30 Mbps. Therefore, the bandwidth of the emulated link #2 remains 50 Mbps. The emulated counter is changed from 0 to 2. If the threshold was decided as two, the emulated link delay is increased from 8 to 10 (The incremental value is also decided by network administrator).

To search the optimal path, a simple parallel shortest path search was used. When a new traffic demand comes, an EPSP is broadcasted from the source router to each branch. The bandwidth of a passed link is recorded in the bandwidth field of the EPSP. When the EPSP arrived at a new router, an EPSP is broadcasted again. If an EPSP arrived at a new link which has smaller bandwidth than the bandwidth which is recorded in the EPSP, the smaller bandwidth value is rewritten into the bandwidth field. The index number of the passed link is also recorded in EPSPs. Finally, EPSPs are collected in the destination router. The faster arrived EPSP indicates the path which has smaller delay. Since the smallest bandwidth along the path is recorded in the EPSP, we can choose the optimal path which has enough bandwidth and small delay for each traffic demand.

Figure 6 shows an example of bandwidth recording. In this example, a EPSP arrived at the emulated link #2. The bandwidth which recorded in the EPSP is 80 and it is bigger than the bandwidth of emulated link #2. Then, the bandwidth value of the emulated link #2 is rewritten into the EPSP. The bandwidth value of the emulated link #3 is 90, and it is bigger than the bandwidth which recording in the EPSP. Therefore, When the EPSP arrived at the emulated link #3, the bandwidth value which recording in the EPSP is not modified.

As an important step of our method, we proposed a simple parallel shortest path search algorithm using an emulated network on DRP. We send an EPSP and let this EPSP pass
through each path between a source router and a destination router. The packet which passes the shortest path will arrive at the destination node first. The summary of this algorithm is shown as follows.

step 1 Assign index numbers to all links and routers.
step 2 Send an emulated packet from a source router, and broadcast it to each branch of the source node.
step 3 When the emulated packet passes through a link, the index number of the link is recorded in the emulated packet. When the emulated packet arrived at a neighboring router, the emulated packet is broadcasted.
step 4 Repeat step 3 until an emulated packet arrived at the destination node. The information of the first arrived emulated packet includes the shortest path.

IV. PROTOTYPE IMPLEMENTATION OF THE PROPOSED ROUTING ENGINE

We construct an emulated network on a commercially available DRP, DAPDNA-2, developed by IPFlex Inc [19], [20]. DAPDNA-2 consists of Digital Application Processor (DAP), a high-performance RISC core, and Distributed Network Architecture (DNA). The DNA is embedded in an array of 376 Processing Elements (PEs), which are comprised of computation units, memory, synchronizers and counters. The DNA has several 4 memory banks to store configurations. Only 1 memory bank is active while execution, and the other is for background storage. DNA can change configuration by loading another configuration among 3 background memories.

An emulated network is constructed with several PEs. We set parameters of each PE to emulate various functions which real routers and links have. For searching the route from the source node to the destination node, we send an emulated packet which transmits on DRP. Figure 4 is an example of our emulated network construction. There are 6 routers and 10 links in the real network, so we construct 6 emulated routers and 10 emulated links by using several PEs and connect between these emulated routers and links. The word size of DAPDNA-2 is 32 bits. Then we read a 32-bit data as a EPSP from the main memory of DAPDNA-2, and transmit it through the emulated network. The emulated packets are broadcasted from the source router. Finally, we collect EPSP at the destination router and write it into the memory. When we want to use this EPSP, we can read it from the memory.

Our proposed algorithm is implemented on the evaluation board, DPADNA-EB4. DPADNA-EB4 is a full-size PCI board and plugged into a PCI slot of the PC as shown in Figure 7. There are two DAPDNA-2 processors on a DPADNA-EB4.

A. Implementation of shortest path search

In this implementation of shortest path search, we define the bitmap of the EPSP and the link delay is fixed. This time, bandwidth is not considered in the implementation. The design of the virtual node and virtual link is also described in this subsection.

We first describe the bitmap of the EPSP. Figure 8 shows an example of the bitmap of EPSP.

In this example, a 32-bit EPSP is divided into two fields. The lower 26 bits of EPSP is a field where the index numbers of links are recorded. The upper 6 bits is a field that the bandwidth of the passed route is recorded. However, this field is not used in this time.

The bitmap of an EPSP in this example can be used in a network which has 26 links. If network expand, we should use more EPSP to collect path information. For example, if we want to implement our proposed algorithm on a network which has 52 links, we can use two 32 bits EPSPs to collect path information. (We define this two packets is a one set.) At every branch point, we broadcast the set of emulated packets.

The functions of an emulated router are shown as follows.

- Copy the EPSP from the input port.
- Send the EPSP to other output ports except for the output port corresponding to the input port.
- Avoid packet contentsion. It means to prevent EPSPs from conflicting when two or more EPSPs arrived in an emulated router at the same time.

The functions of an emulated link are shown as follows.

- Record the index number of the link into the arrived emulated packet.
- Prevent emulated packets from loop. It is done to check the bitmap of the emulated packet.
- Generate delay corresponding to the link cost of the real link. The unit of delay is expressed as a clock cycle of DAPDNA-2.

Figure 9 shows an example of our shortest path search on DAPDNA-2. There are 6 routers and 9 links in Fig. 9. First, we assign index numbers of all routers and links. In the example, all 32 bits of a EPSP is used for recording the information of the index numbers of links. However, only 9 bits are shown in the example because there are only 9 links. An EPSP is initialized with 000000000.

step 1 At the source router, we broadcast emulated packets
Fig. 9. Example of bitmap of an EPSP

000000000. The emulated packet will be sent to the link #1 and the link #2.

step 2 The emulated packets pass through the link #1 and #2, and the link number is recorded to the emulated packet. The output of link #1 and link #2 are 000000001 and 000000010 respectively. Then, the node 1 sends 000000001 to the link #3 and #4 in the same way. The output emulated packets of the link #3 and #4 will be 000000101 and 000001001.

step 3 When the EPSP arrived at other nodes, step 2 will be repeated until an EPSP arrives at the destination router.

step 4 Finally, the information of the EPSP first arrived at the destination router shows includes the shortest path information in its field.

In the network shown in Fig. 9, the EPSP first arrived at the destination router is 001000101. The information means the the shortest path passes through the link #1, #3, and #7. As a result, the shortest path is 0-1-2-5.

Our proposed approach also can collect all route information between a source node and a destination node that not only shortest path.

V. EXPERIMENTAL RESULTS

In this section, we implement the shortest path search in our emulated network. We compare with Dijkstra’s algorithm, also compare all routes searching and Breadth first search method. We execute two conventional algorithms using 3 GHz processor of Intel Pentium 4 processor and execute our method using 166 MHz reconfigurable processor of DPADNA-2.

A. Simulation Result of Dijkstra’s algorithm and our proposed

We evaluate the calculation clocks of Dijkstra’s algorithm and our proposed algorithm. The network topology is the same as Figure 4. We execute Dijkstra’s algorithm and our proposed algorithm 100 times and we obtain the average of execution time. The simulation result is shown in Table I.

We can see that our algorithm collects all route information faster than the breadth first search algorithm because our algorithm collects the link information at each route at the same time.

TABLE I

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Execution time (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra’s algorithm</td>
<td>12.8</td>
</tr>
<tr>
<td>Proposed algorithm</td>
<td>0.67</td>
</tr>
</tbody>
</table>

We can see that our algorithm collects all route information faster than the breadth first search algorithm because our algorithm collects the link information at each route at the same time.

B. Simulation result of the Breadth First Search algorithm and our proposed

We compare execution time of collecting all route information between the breadth first search algorithm and our all path pattern search method. The simulation result is shown in Table II.

TABLE II

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Execution time (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth First Search algorithm</td>
<td>3300</td>
</tr>
<tr>
<td>Proposed algorithm</td>
<td>2</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

The power consumption of network is an issue in the future network technologies because it is expected that the percentage of the power consumption of network is increasing in a next decade. Therefore, a new network architecture suitable for improving power efficiency is required. In this paper, a new network architecture, Cloud C-plane, is proposed to aggressively optimize the power efficiency of the network. The routing engine on DRP is a key component in Cloud C-plane to achieve high speed path calculation. We implemented the prototype of the routing engine on DRP, and the performance evaluation about path calculation was conducted. The result shows our prototype routing engine is 19 times faster than Dijkstra’s algorithm.

VII. ACKNOWLEDGEMENTS

This work is supported in part by a Grant-in-Aid for the Global Center of Excellence for high-Level Global Cooperation for Leading-Edge Platform on Access Spaces from the Ministry of Education, Culture, Sport, Science, and Technology in Japan, and Grant-in-Aid for Scientific Research (19360178). One of the authors, Sho Shimizu, appreciate Yoshida Scholarship Foundation for its financial support.
REFERENCES


