A High-Speed Routing Engine for Software Defined Network

Shan Gao, Sho Shimizu, Satoru Okamoto and Naoaki Yamanaka

Abstract—**Recently, attention is particularly focused on the research of Software defined network (SDN) for reducing network management complexity. The one of a key technology of SDN is OpenFlow. OpenFlow provide a centralized controller for network and the scalability of controller is main issue. In this paper, we propose a high-speed routing engine for improve the scalability of OpenFlow controller. Unlike conventional architectures of routing engine, the proposal is a hardware routing engine that using on-chip diorama network. We define the Diorama Network as a virtual emulated network in a chip. We implement a prototype of the routing engine on an actual dynamically reconfigurable processor (DRP), and test results show that the prototype can execute the shortest path calculation 19 times faster than the current approach.**

Index Terms—**Software-Defined Network, OpenFlow, Routing Engine, Dynamically Reconfigurable Processor (DRP)**

I. Introduction

Today, the Internet is becoming the key global infrastructure for telecommunication. The rapid adoption of the Internet is promoting the growth of the world economy and globalization. The Internet traffic is rapidly increasing due to the increasing number of users and their use of higher bandwidth services. Therefore, the cost and complexity of network management becomes a challenging problem. Software-defined network [1] becomes the most remarkable approach to network traffic control.

The SDN architecture decouples the forwarding plane and control plane of network device such as router or switch, and runs control plane in software. Decoupling makes the network more advanced since the speeds at which their technologies evolve are different. OpenFlow [2-3] is the key technology of SDN, because OpenFlow can provide interoperability and better performance to SDNs. Network Operators could define traffic flows and determine how packets are forwarded through switches or routers over a network using a remote OpenFlow controller. The remote OpenFlow controller can communicate OpenFlow switch by OpenFlow protocol via a secure channel. OpenFlow Controller is programmable, Service Provider can provide their unique services very convenient, also can implement the traffic engineering and management method faster.

OpenFlow networks have been implemented on some university campuses in US [4]. The large scale OpenFlow based network is also researched. The Global Environment for Network Innovations (GENI) project has just start to applying OpenFlow in its network infrastructure [2], [6]. In [5], a nation-wide OpenFlow based network on the NICT JGN2plus testbed is deployed. Therefore the OpenFlow is not only researched for a campus network, also for the large-scale network. The OpenFlow network controller is centralized control node for one a network. Therefore, the scalability and reliability becomes key issues of controller. A data center that has 100 edge switches, the centralized controller can expect to see about 10 million flow requests per second [7]. When network is large and traffic is heavy, routing becomes a challenging problem of OpenFlow controller, since bed routing speed will increase the response speed of controller for each OpenFlow switch in forwarding plane and leading bed performance of OpenFlow network.

In this paper, we propose a high speed routing engine and establishing a prototype of routing engine on a Dynamically Reconfigurable Processor (DRP). This approach makes use of an on-chip emulated network that is called diorama network. Emulated packets are transmitted throughout the emulated network, and the shortest path is identified because the first emulated packet from the source node to the destination indicates the shortest path. We develop a prototype of the routing engine on an actual DRP.

This paper as organized as follows. Section Ⅱ describes the architecture of OpenFlow network. In Section Ⅲ, we explain the basic algorithm of the proposed routing engine as implemented on a DRP. The prototype of the routing engine is shown in Section Ⅳ, and test results are provided in Section Ⅴ. Finally, we summarize this paper in Section Ⅵ.

II. OpenFlow Architecture

The current router consists of two main functions; forwarding and control [8]. SDN uses the terms forwarding element and control element to refer to blocks that offer forwarding functions and control functions, respectively. The control element of the current router corresponds to its operating system, such as IOS [9], JUNOS [10], OpenFlow. The

Shan Gao, Sho Shimizu, and Naoaki Yamanaka are with the Yamanaka Laboratory,
Department of Information and Computer Science,
Keio University, 3-14-1
Hiyoshi, Kohoku, Yokohama, Kanagawa, JAPAN 223-8522
(e-mail:shan.gao@yamanaka.ics.keio.ac.jp)

forwarding element and control element are tightly coupled in the current router as shown in the left side of Fig. 1. SDN architecture, on the other hand, decouples them as shown in the right side of Fig. 1. Since the forwarding element and control element have different rates of evolution, decoupling is advantageous because they can be advanced independently.

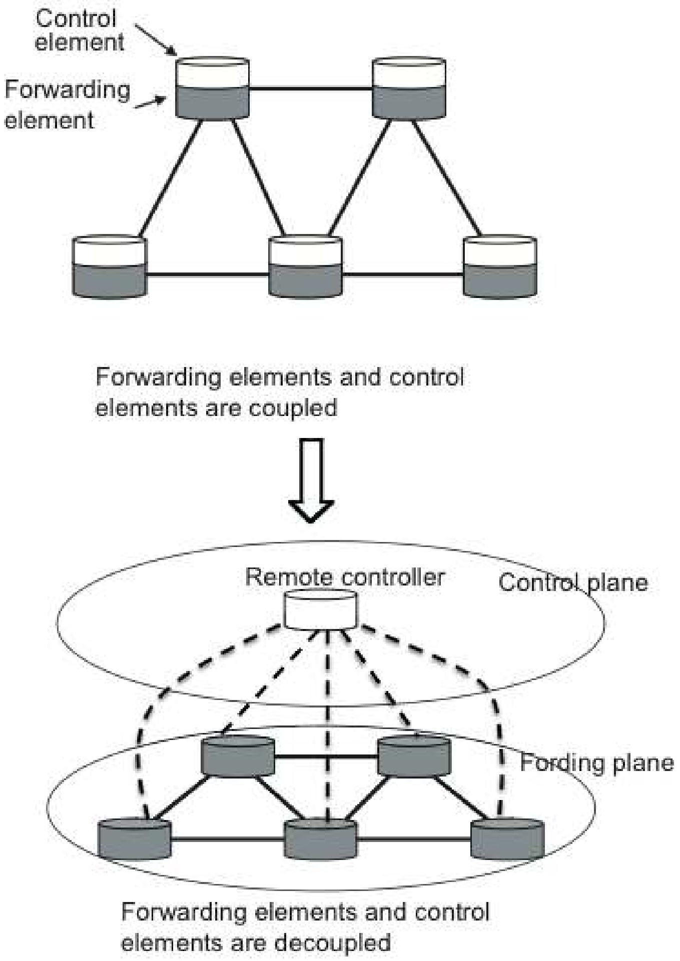
Control element
Forwarding element
Forwarding elements and control elements are coupled
Remote controller
Control plane
Fording plane
Forwarding elements and control elements are decoupled

Fig. 1. Forwarding element and control element are decoupled in Software –Defined network

The controller is controlling and managing the tasks of its corresponding forwarding element such as routing. A control element communicates with its forwarding element by the Forwarding Element Control Protocol (FECP), such as GSMP [11] or OpenFlow Protocol [12]. It is an interface between forwarding elements and control elements. For example, a control element sets the forwarding configuration via FECP. Across the network, the control elements form the control plane, and the forwarding elements the forwarding plane.

A control element does not have to be co-sited with its forwarding element. Fig 2 shows that the control plane lies in the remote place. Control elements can be virtualized as a software service because they are physically decoupled from their forwarding elements. As a result, control elements run on virtual machines, and are likely to be placed in a server in a data center or central office of service provider.

The OpenFlow architecture is the key technology of SDN. OpenFlow based networks have three main parts: OpenFlow controller, OpenFlow switch and OpenFlow protocol. Service provider or user can program OpenFlow controller. Several open source platforms such as NOX [13-14], Trema [15], Beacon/Floodlight [16-17], is provided for develop controller. OpenFlow switch has tow types: software switch such as Open vSwitch [18] and hardware switch such as NEC UNIVERGE PE5240/PF5820, IBM RacSwitch G8264 and HP 3500/5400/8200. OpenFlow protocol provides the handshake function, sending control command message, reporting switch status and so on.

Figure 2 show our network architecture. The on-chip routing engine is the high-speed engine for calculate route for every traffic flow. With different services, the physical network can be virtualized as a virtual network that is called slice. In our approach, different slice is used for different service during routing. Fig 3 shows an example. The OpenFlow controller can maintain these two slices, and provide the optimal flow controlling in these two topologies.

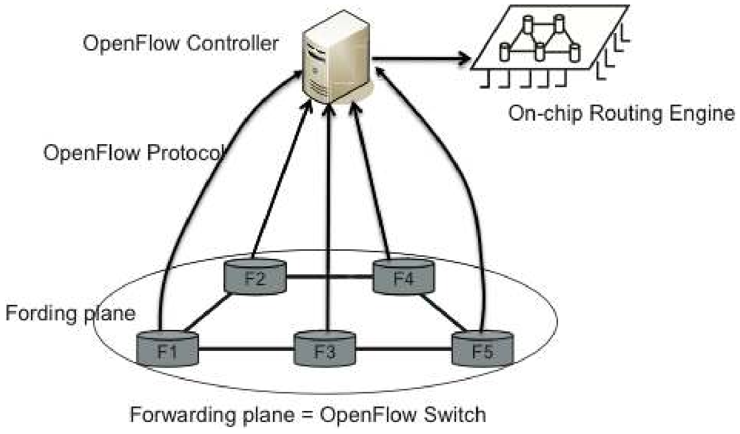

Fig. 2. OpenFlow Based network architecture

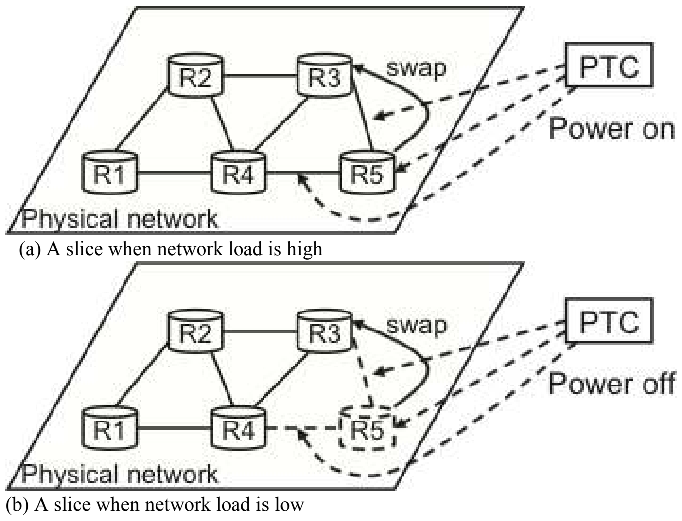

Fig. 3. Several virtual networks in OpenFlow Controller

We propose that the routing engine be based on a Dynamically Reconfigurable Processor (DRP). The architecture of the proposed routing engine is described in detail in the next section.

III. Routing Engine on DRP

The recently advances in the performance of reconfigurable devices, such as Field Programmable Gate Array (FPGA) and Dynamically Reconfigurable Processor (DRP), has been significant [19]. We can design dedicated hardware with sophisticated functions by using these types of devices. They are very attractive since they combine high performance, due to their hardware implementation, with the ability to dynamically alter their internal circuit at high speed, for example, within a few clock cycles. Our routing engine takes full advantage of the dynamic reconfigurability of DRPs.

Our proposal is to make an on-chip emulated network that corresponds to the real network. We transmit emulated packets through the emulated network and observe the behavior of the emulated links and routers on the emulated. That is, we can experimentally optimize the network. Fig. 4 shows a real network and its emulated twin on a DRP.

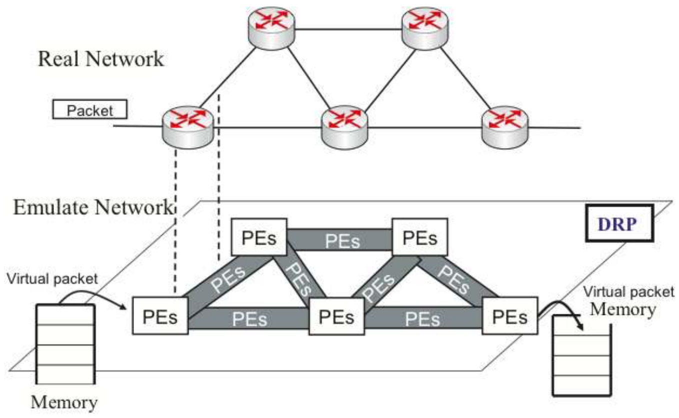

Fig. 4. The emulated network, which corresponds to the real network, is constructed on DRP.

Two types of emulated packet are defined. The first type is called the emulated flooding packet (EFP). An EFP has three main fields: the first field holds the index number of the source router, the second field the bandwidth of traffic demand from the source router, and the third field is the index number of a link.

Each emulated router can get global information from the EFPs sent over the emulated network. The second type is called the emulated path search packet (EPSP). An EPSP has two main fields: bandwidth recording field and link index number recording field. The former is used to record the smallest of the links' bandwidths along the path. The latter is used to record the index numbers of all links passed.

The metrics of delay, link utilization and bandwidth, are considered in the TE method that is run on the emulated network. With regard to link utilization, we can make the link cost change dynamically according to the number of passed packets. For example, a packet count function can be added to the emulated links. When the number of passed packets surpasses a threshold set by the network administrator, the link cost is increased. Therefore, this link will not be chosen when next searching for the shortest path. Bandwidth is an emulated link parameter and both EFPs and EPSPs have a field to record the bandwidth of each link passed. Each emulated link generates delay of several clock cycles according to the real link cost. Fig. 5 shows an example of parameter initialization by using EFPs. In this example, the paths available to the two EFPs are abbreviated. One EFP is from emulated node X and its bandwidth is 80 Mbps. The other EFP is from emulated node Y and its bandwidth is 30 Mbps. Therefore, the bandwidth of the emulated link #2 remains 50 Mbps. The emulated packet counter of link #2 is changed from 0 to 2. If the threshold was set at two, the emulated link delay is increased from 8 to 10 (The step value is also set by the network administrator).

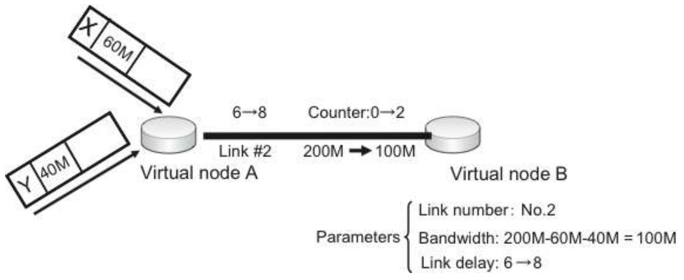

Fig. 5. Parameter deterministic method on emulated links

The optimal path is located by conducting a simple parallel shortest path search. When a new traffic demand arises, an EPSP is broadcast from the source router to each branch. The bandwidths of passed links are recorded in the bandwidth field of the EPSP. When the EPSP arrives at a new router, it is rebroadcasted. If the EPSP arrives at a new link that has smaller bandwidth than the value recorded in bandwidth field of the EPSP, the smaller bandwidth value is written into the bandwidth field. The index number of the passed link is also recorded in the EPSP. Finally, EPSPs are collected at the destination router. The EPSP that arrives first identifies the path that has the smallest delay. Since the smallest bandwidth along the path is also recorded in the EPSP, we can choose the optimal path that has enough bandwidth and acceptable delay for each traffic demand.

Fig. 6 shows an example of bandwidth recording. In this example, an EPSP arrives at emulated link ¥#2. The bandwidth recorded in the EPSP is 80, which is larger than the bandwidth of emulated link #2, 50. Thus 80 is replaced by 50 in the EPSP. The bandwidth value of emulated link #3 is 90 which is larger than 50, and so the value of 50 is not replaced.

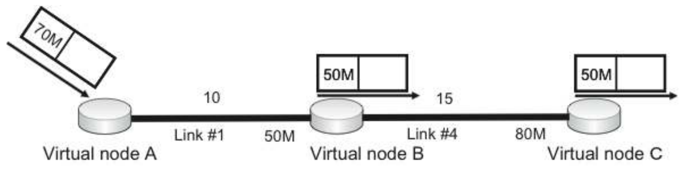

Fig. 6. Link bandwidth recording method using emulated links

Our simple parallel shortest path search algorithm sends an

EPSP across each path between one source router and one destination router. The packet that transits the shortest path will arrive at the destination node first. This algorithm is summarized as follows.

Step1 Assign index numbers to all links and routers.

Step2 The source router issues an emulated packet and broadcasts it to each branch known to the source node.

Step3 When the emulated packet passes through a link, the index number of the link is recorded in the emulated packet. When the emulated packet arrives at a neighboring router, the emulated packet is rebroadcasted over all outgoing links except the mirror of the incoming link.

Step 4 Repeat step 3 until the first emulated packet arrives 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 constructed an emulated network on a commercially available DRP, DAPDNA-2, developed by IPFlex Inc [20-21]. DAPDNA-2 consists of a 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 4 memory banks to store configurations. Only 1 memory bank is active while the others are for background storage. DNA can change the network's configuration by loading one of the three stored configurations in background memory.

An emulated network is constructed by linking emulated nodes, each of which consists of several PEs. We set the parameters of each PE to emulate the various functions possessed by real routers and links. Figure ¥ref{fig:emulated_network} 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 connect these emulated routers and links. The word size of DAPDNA-2 is 32 bits, so we read 32-bit data as an 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 the first EPSP at the destination router and write it into memory where it can be accessed for later use. Fig 7 shows an example of designing a virtual node. To replicate a degree-3 node, we use three PEs to construct a virtual node that has 3 input ports and 3 output ports.

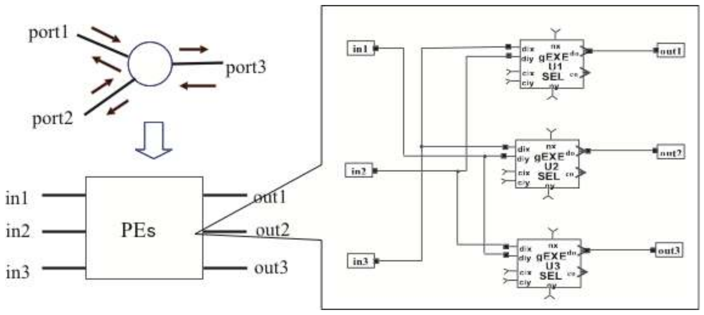

Fig. 7. Example of virtual node structure having node degree of three

Our algorithm was implemented on the evaluation board, DPADNA-EB4. DPADNA-EB4 is a full-size PCI board as shown in Fig 8 and is simply plugged into a PCI slot of the PC. There are two DAPDNA-2 processors on each DPADNA-EB4.

Fig. 8. The evaluation board, DPADNA-EB4

A. Implementation of shortest path search

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

We first describe the bitmap of the EPSP, see Fig. 7. In this example, a 32-bit EPSP is divided into two fields. The lower 26 bits are link index number field. The upper 6 bits is the link bandwidth field but this field is not used in this implementation.

This EPSP bitmap supports networks with up to 26 links. To handle a network that has 52 links, we would simply use two 32 bits EPSPs to collect path information. (We define these two packets as one set.) Every branch point rebroadcasts the set of emulated packets.

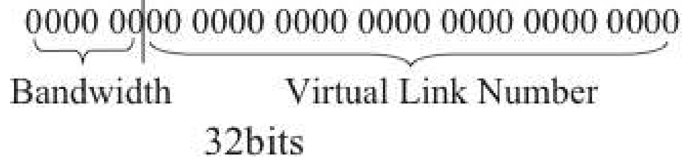

Fig. 9 Example of bitmap of an EPSP

The functions of an emulated router are shown as follows.

- Copy the EPSP from the input port.
- Send the EPSP to all other output ports except the output port that corresponds to the input port.
- Avoid packet contention. This means preventing EPSP

conflict when two or more EPSPs arrived at 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 the looping of emulated packets. This is done by checking the bitmap of the emulated packet.
- Generate a delay corresponding to the link cost of the real link. Delay is expressed in units of the clock cycle of DAPDNA-2.

Fig. 11 shows an example of our shortest path search on DAPDNA-2. There are 6 routers and 9 links in Fig. 11. First, we assign index numbers to all routers and links. In the example, all 32 bits of an EPSP are used for recording the index numbers of links; 9 bits are shown in the example because there are only 9 links. Each EPSP is initialized with 000000000.

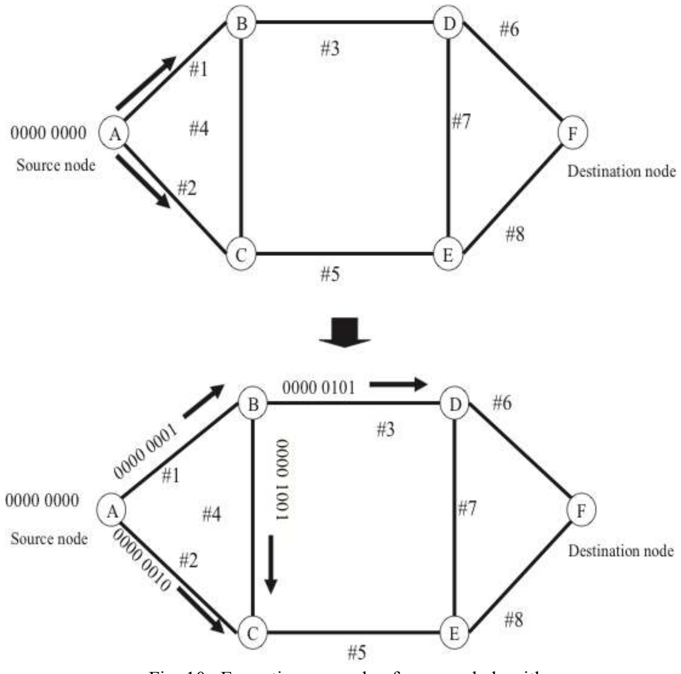

Fig. 10. Execution example of proposed algorithm

Step 1 At the source router, we broadcast the emulated packet containing 000000000. The emulated packet is passed to link ¥#1 and link ¥#2.

Step 2 The emulated packets pass through link ¥#1 and ¥#2, and the link numbers are recorded in the emulated packets. The outputs of link ¥#1 and link ¥#2 are 000000001 and 000000010, respectively. Next, node 1 sends 000000001 to link ¥#3 and ¥#4 in the same way. The output emulated packets of link ¥#3 and ¥#4 are 000000101 and 000001001.

Step 3 When the EPSP arrives at other nodes; step 2 is repeated until the first EPSP arrives at the destination router.

Step 4 Finally, the shortest path information is determined from the contents of the first EPSP.

In the network shown in Fig. 11, the first EPSP to arrive at the destination router holds 11000101. This means that the shortest path is formed by links #1, #3, #7 and #8. As a result, the shortest path is A-B-D-E-F. Our proposed approach also can collect all route information between a source node and a destination node, i.e. not just the shortest path.

B. Change configuration dynamically

| DAPDAN-2 has three internal memory banks. Therefore, we can store three network topologies as a hardware configuration in DAPDNA2. If the configuration is stored in internal memory, DAPDAN-2 can change the configuration only in few seconds. Fig. 12 shows an example. In this example, three different network topologies are stored in memory bank.

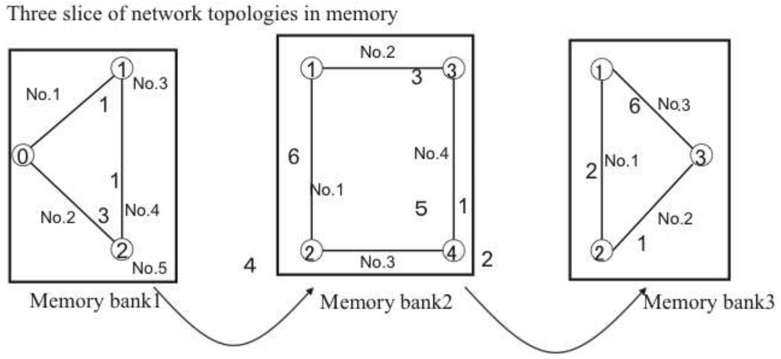

Fig. 11. Three slice in DAPDNA-2

V. Experimental Results

In this section, we compare the shortest path calculation time between proposed method and the method that used in the current routing system. We implement the shortest path search in our emulated network. We compare it to Dijkstra's algorithm, as well as Breadth first search method. The current path search methods are implemented as C applications and we ran them on a 3 GHz Intel Pentium 4 processor. Our method is executed on the 166 MHz reconfigurable processor of DPADNA-2.

TABLE I
EXECUTION TIME OF DIJKSTRA'S ALGORITHM AND OUR PROPOSED ALGORITHM

Execution time: (μ s)	
Dijkstra's algorithm	Proposed Algorithm

A. Comparison result of shortest path calculation

We measured the calculation clocks of Dijkstra's algorithm and our proposed algorithm. The network topology is the same as that in Figure 4. We executed Dijstra's algorithm and our proposed algorithm 100 times and averaged the execution times. The simulation results are shown in Table 1.

The execution time of our algorithm is faster than the execution time of Dijkstra's algorithm because our algorithm is executed in parallel. However, the Dijkstra's algorithm runs serially. Therefore, under the same conditions, our algorithm runs faster than Dijkstra's algorithm. Additionally, the calculation time of Dijkstra's algorithm will increase in a larger network. The calculation time of our algorithm only depends on the total cost of the shortest path. Therefore, our algorithm can search shortest path faster even if the network is large.

B. Comparison result of the all paths calculation

In this sub section, we compared the all paths calculation time. In this paper, all paths search is to find paths connecting

TABLE II
EXECUTION TIME OF THE BREADTH FIRST SEARCH ALGORITHM AND OUR PROPOSED ALGORITHM

Execution time: (μ s)	
Breadth First Search	Proposed method
3500	2

the given source node to anywhere in the network. For example, the source is node a fig.10. All paths search is to list all shortest path between node A and other nodes that includes node B, node C, node D, node E and node F. We measured the execution time taken by the breadth first search algorithm and our all path pattern search method to collect all route information. The results are shown in Table 2.

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

VI. CONCLUSION

The SDN approach is a useful solution to a lot of current network problem such as management. OpenFlow is a key technology to realize SDN and the scalability of Openflow controller is one of a major issue. When the controller manage a large network, routing speed is becomes a problem. We challenged the scalability of the OpenFlow controller. In this paper, we proposed an on-chip routing engine allows OpenFlow controller to achieve high-speed path calculation. We implemented a prototype of the routing engine on a DRP, and performance evaluations on path calculation were conducted. The results show that our prototype routing engine is 19 times faster than the current shortest path search method that is Dijkstra's algorithm. Therefore our proposed system can improve the routing speed of OpenFlow controller and enable high scalability of OpenFlow controller.

ACKNOWLEDGMENT

This work is partly supported by the National Institute of Information and Commucations Technology (NICT).

REFERENCES

[1] Goth, G. "Software-Defined Networking Could Shake Up More than Packets," Internet Computing, IEEE, Vol 15, Issue 4, pp.6-9, 2011
[2] N/McKeown, T.Anderson, H. Balakrishnam, G. Parulkar, L. Peterson, J.Rexford, S.Shenker, and J.Turner, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, Vol.38, Issue 2, no. 2, pp. 69-74, 2008.
[3] Steven J., Vaughan-Nichols, "OpenFlow: The Next Generation of the Network?" Computer, Vol. 44, Issue 8, pp.13-15, 2011
[4] Kanaumi, Y.; Saito, S.; Kawai, E. "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area OpenFlow-based network," Network and Service Management (CNSM), pp.330-333, 2010.
[5] Shimonishi, H.; Ishii, S.," Virtualized network infrastructure using OpenFlow, " Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP, pp.74 – 79, 2010.
[6] "GENI," http://www.geni.net/.
[7] Zheng Cai, Allan L. Cox, T.S. EugeneNg, " Maestro: A system for Scalable OpenFlow Control", Rice University Technical Report TR10-08, December 2010
[8] Ramjee, R.; Ansari, F.; Havemann, M.; Lakshman, T.V.; Nandagopal, T.; Sabnani, K.; Woo, T., "Separating Control Software from Routers," Communication System Software and Middleware, pp.1-10, 2006.
[9] Cisco IOS, http://www.cisco.com/en/US/products/ps6537/products_ios_sub_category_home.html
[10] Juniper Networks, http://www.juniper.net/jp/jp/products-services/nos/junos/
[11] A.Doria, F.Hellstrand, K.Sundell, and T.Worster, "General switch management protocol (GSMP) v3," Request For Comments (RFC), no.3292, Jun. 2002.
[12] B.Heller, "OpenFlow switch specification version 0.8.9," http://www.openflow.org/documents/openflow-spec-v0.8.9.pdf
[13] N.Gude, T.Koponen, J.Pettit. B.Pfaff, M. Casado, N. McKeown and S. Shenker, "NOX: towards an operating system for networks." SIGCOMM Computer Communication Rev., vol. 38, pp. 105-110, July 2008.
[14] NOX, http://www.noxrepo.org/
[15] Trema, http://trema.github.com/trema
[16] Beacon, http://www.beaconcontroller.net/
[17] Floodlight, http://floodlight.openflowhub.org/
[18] Open vSwitch, http://openvswitch.org/
[19] H.Amano,"A survey on dynamically reconfigurable processors," IEICE Transactions on Communications, vol. E89-B, no.12, pp.3179-3187, Dec.2006
[20] T.Sugawara, K.Ide, and T.Sato, "Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology," IEICE Transactions on Information and Systems, vol. E87-D, no.8, pp.1997-2003, Aug.2004.
[21] *" Dynamically reconfigurable processor, DAPDNA-2," http://www.tokyo-keiki.co.jp/hyd/e/products/20120508_dap01.html, 2010..*

Shan Gao received B.E. and M.E degrees from Keio University, Japan, in 200in Graduate School of Science a8 and 2010, respectively. He is currently working toward the Ph.D. degree in Graduate school of Science and Technology, Keio University, Japan. Since 2010, he has been researching about the network architecture and traffic engineering on the next generation optical network. In 2010, he will became a research assistant of Keio University Global COE Program, ``High-level Global Cooperation for Leading-edge Platform on Access Spaces" by Ministry of Education, Culture, Sports, Science and Technology, Japan. He is a student member of the IEEE of Japan.

Satoru Okamoto received the B.E. and M.E. degrees from Keio University, Japan, in 2005 and 2007, respectively. He is currently working toward the Ph.D. degree in Graduate School of Science and Technology, Keio University, Japan. His research interests include network architecture and traffic engineering on the next generation optical network. In 2007, he became a research assistant of Keio University Global COE Program, ``High-level Global Cooperation for Leading-edge Platform on

Access Spaces" by Ministry of Education, Culture, Sports, Science and Technology, Japan. He is a student member of the IEEE, the OSA, and the IEICE. Satoru Okamoto received the B.S., M.S, and Ph.D. degrees in electronics engineering from Hokkaido University, Hokkaido, Japan in 1986, 1988 and 1994 respectively. In 1998, he joined Nippon Telegraph and telephone Corporation (NTT), Japan. Here, he engaged in research on ATM cross-connect system architectures, photonic switching system, optical path network architectures, and developed GMPLS controlled HIKARI router (Photonic MPLS router) systems. He leads several GMPLS related interoperability trials in Japan, such as the Photonic Internet Lab (PIL), OIF worldwide interoperability demo, and Keihanna Interoperability Working Group. From 2006, he has been an Associate Professor of Keio University. He is a vice co-chair of Interoperability Working Group of Kei-han-na Info-communication Open Laboratory. He is now promoting several research projects in the photonic network area. He received the young Researchers' Award and the Achievement Award in 1995 and 2000, respectively. He has also received the IEICE/IEEE HPSR2002 outstanding paper award. He is associate editor of the IEICE transactions and the OSA Optics Express. He is an IEEE Senior Member and an IEICE Fellow.

Naoaki Yamanaka graduated from Keio University, Japan where he received B.E., M.E., and Ph. D. degrees in engineering in 1981, 1983 and 1991, respectively. In 1983 he joined Nippon Telegraph and Telephone Corporation's (NTT's) Communication Switching Laboratories, Tokyo, Japan, where he was engaged in research and development of a high-speed switching system and high-speed switching technologies for Broadband ISDN services. Since 1994, he has been active in the development of ATM base backbone network and system including Tb/s electrical/Optical backbone switching as NTT's Distinguished Technical Member. He is now researching future optical IP network, and optical MPLS router system. He is currently a professor of Keio Univ. and representative of Photonic Internet Lab. (PIL). He has published over 126 peer-reviewed journal and transaction articles, written 107 international conference papers, and been awarded 182 patents including 21 international patents. Dr. Yamanaka received Best of Conference Awards from the 40th, 44th, and 48th IEEE Electronic Components and Technology Conference in 1990, 1994 and 1998, TELECOM System Technology Prize from the Telecommunications Advancement Foundation in 1994, IEEE CPMT Transactions Part B: Best Transactions Paper Award in 1996 and IEICE Transaction Paper Award in 1999. Dr. Yamanaka is Technical Editor of IEEE Communication Magazine, Broadband Network Area Editor of IEEE Communication Surveys, and was Editor of IEICE Transaction as well as vice director of Asia Pacific Board at IEEE Communications Society. Dr. Yamanaka is an IEEE Fellow.

Sho Shimizu received the B.E., M.E. and Ph.D degrees from Keio University, Japan, in 2005, 2007 and 2010 respectively. He is currently working in FUJITSU Lab, Japan. His research interests include network architecture and traffic engineering on the next generation optical network. In 2007, he became a research assistant of Keio University Global COE Program, ``High-level Global Cooperation for Leading-edge Platform on Access Spaces" by Ministry of Education, Culture, Sports, Science and Technology, Japan. He is a member of the IEEE, the OSA, and the IEICE.