Efficient Multicasting Algorithm Using SDN

Alaa M. Allakany[1] and Koji Okamura[2]

[1] Graduate school of Information Science and Electrical Engineering, Kyushu University. Japan.
& Faculty of Science, Kafrelsheikh University, Kafrelsheikh, Egypt

[2] Research Institute for Information Technology, Kyushu University. Japan.

Abstract

Many group communication applications require multipoint communication, in order to reduce network traffic rates. Multicast technology has been used as an efficient and scalable technology for data distribution. However, in IP network, the responsibility for management of multicast groups is distributed among network routers and routing rules calculated based on local view resulting from sharing neighbor's routers its information. Distributed calculation and local view of IP network causes some limitations, such as delays resulting from processing group events, and failed for calculating optimum solutions that required a global view of the network. Software Defined Networking (SDN) represented by OpenFlow presented as a solution for many problems, in SDN the control plane and data plane are separated by shifting the control and management to a remote centralized controller with a global view of the network, and the routers are used as a forwarder only. Recently, several researchers have been proposed multicast routing algorithms for solving the problem of shortest path tree (SPT) and Minimum Steiner tree (MST) in SDN. SPT can calculate multicast tree faster than MST. However, MST can generate solutions optimum than SPT.

In this paper we take the advantage of OpenFlow to propose and implement multicasting OpenFlow controller, this centralized controller is a core part of our multicasting approach. For constructing the multicast tree, we proposed a new algorithm that combines both of Dijkstra shortest path algorithm and Tabu search (TS) algorithm. In the proposed algorithm, Dijkstra algorithm and TS work respectively for fast start-up multicast session and optimum solution. Proposed algorithm take the advantages of both algorithms such as fast convergence time of Dijkstra algorithm and optimum solution of TS and avoid the shortages of both algorithms. We validate our approach using the popular Mininet network emulation environment with Pox controller. The results prove that our approach can improve start-up time for initialization multicast session. Also, can minimize constructed the multicast tree.

Key words:

Software Defined Networks, OpenFlow, Pox controller, Mininet, Multicast tree, Dijkstra Algorithm, Tabu Search (TS).

1. Introduction

IP multicast is a distribution paradigm that sends IP packets to multiple receivers in a single transmission, in a one-to-many or many-to-many fashion. This allows reducing source server load and increasing network capacity savings. IP multicast supports a variety of applications such as IPTV streaming, video conferencing, multi-location backups or online multi-player gaming [1]. IP multicasting still faces some problems: Firstly, Traditional multicast routing algorithms require routers to participate in data forwarding and control management. Then multicast routers need to maintain each group state, which arouses a lot of control overhead and add substantial complexity to routers. Secondly, Routers construct and update the multicast tree in a distributed manner; each router has only local or partial information on the network topology and group membership and there are high number of communication messages that neighboring routers have to exchange in order to update their multicast trees at every time a client joins or leaves a multicast group. These cause more latency time and difficulty to build an efficient multicast tree due to the lack of global information.

Recently, SDN is presented as a networking approach that facilitates the decoupling of the control plane in a network using a remote controller from the data plane. OpenFlow protocol [2], defines the communication between OpenFlow switches and the controller of the network. With the centralized network, OpenFlow controller has a global view of the current status of the network and can interact with its network devices. All the multicast management, such as multicast tree computing, group management are handled by this controller, and the controller has complete knowledge of the topology and the members of each group, then it can create more efficient multicast trees than the distributed approach [3].

Various multicast mechanism and algorithm are proposed, the author in [4] provides a mechanism to compute multicast trees centrally by flooding group membership information to all multicast routers. This mechanism (MOSPF) has a scalability problem that all routers have to compute a multicast tree per multicast group when the new multicast group appears or receivers join in or leave from multicast groups. The Protocol Independent Multicast - Sparse Mode (PIM-SM) is the most common for IP multicasting, the routing algorithms of this protocol are not designed to build optimal routing trees [5]. PIM-SM builds

Manuscript received April 5, 2017
Manuscript revised April 20, 2017

trees rooted at either the source of the multicast group or at a pre-determined rendezvous point (RP) for the group.

In [6] the author has suggested high-level primitives (API) based in Open-Flow to provide a more friendly development of multicasting networks. These primitives have a simplified implementation of the OpenFlow multipoint protocol but does not consider questions such as changes in multicast groups. In [7] the author proposed a multicast clean-slate approach logically centralized based on SDN and anticipated processing for all routes from each possible source. The author of this paper aiming to reduce event delays from source to each destination and don't consider minimizing the total edges in construction multicast tree. Finally, in [8] this paper proposed a novel multicast mechanism based on OpenFlow to separate the data and control plane by shifting the multicast management to a remote centralized controller. The Dijkstra algorithm is used to construct spanning tree in the network and after that drive the multicast tree from existing spanning tree. This method can't construct optimum multicast tree because MST using Dijkstra algorithm can construct a tree with shortest path from source to every destination in the network but it can't minimize the multicast tree.

Optimal tree building is equivalent to solving the Steiner Tree [9] problem, this problem is known to be NP-complete. Some of the multicasting algorithms that used for solving this problem required centralized computation. Heuristic algorithms are efficient for solving this problem but it assumes centralized computation, so implementing these algorithms is not efficient in IP multicast network. OpenFlow enables us to implement these algorithms because of centralized control and programmability. Tabu search will be used as one of heuristic algorithms with Dijkstra algorithm for construction the multicast tree and maintain a multicast tree in case of join in or leave any members from multicast group.

The objective of this work is proposing multicasting OpenFlow controller's modules. For multicasting rules required to the multicast session, we proposed a new algorithm based on two individual algorithms, Dijkstra algorithm and heuristic Tabu Search algorithm. These two algorithm work respectively to solve some multicast routing problems. In our method, the proposed multicast routing algorithm take the advantage of Shortest Path Tree algorithms represented by Dijkstra algorithm and Minimum Sterner Tree algorithms represented by Tabu Search algorithm and avoid the shortages of both. We design SDN controller based on POX controller to achieve the following goals:

1- Implementing multicasting in SDN network with using the features of OpenFlow protocol.
2- Proposed a module in this controller to constructing the multicast tree based on our routing algorithm.
3- Reduce latency time required for initialization multicast tree.
4- Construction near an optimum multicast tree (minimizing the size of the multicast tree).

The details of design our controller modules will discuss in following sections. This paper is organized as follows. In Section 2, we present the design of our multicasting OpenFlow controller and give details of functions of each module in this controller. Then in section 3, we present implementation of our OpenFlow controller modules, we show how Tabu Search algorithm as a heuristic algorithm with Dijkstra algorithm can find an efficient solution for solving multicasting problems. In section 4, we show the evaluation of our algorithm. Finally, we describe the conclusion in Section 5.

2. Design of Multicasting Controller

In this section, we present the design of our multicasting OpenFlow controller. Fig. 1 show the architecture of our proposed scheme, there are two main components in this architecture, controller and OpenFlow switches (forwarder). The functions of the controller are to manage the multicast group state, construct the multicast tree and handle the host requests sent from the forwarders, then set up flow entries that are required to deliver multicast packets into the switches. While forwarders only need to receive instructions from the controller and forward data. Our controller consists of four modules to implement our proposed algorithm, the details and the function of each module in this controller as follow.

- ***Topology discover module:*** this module uses Link Layer Discovery Protocol (LLDP) to discover network topology. By using the information resulting from this module we can build up the network topology graph G(V, E), where the node set V corresponds to the switches and the edge set E corresponds to the links. Then, we send the data relative to the topology graph G to tree construction module to build up the tree.

- ***Multicast groups management module***: This module is responsible for maintain multicast group state by storing sender information including locations of devices and watching IGMP packets from devices and stores the receivers' locations. Then provide this information for others openflow controller modules to construct multicast tree and management multicast

group events (join in or leave any member form current multicast group).

- ***Tree construction module:*** When controller received new request to initialize a multicasting session, firstly multicast group management module process these messages to obtain sender and receiver information, then notify this module for construction the multicast tree. In this module, we use Dijkstra shortest path algorithm for constructing initial multicast tree from source to receivers as described in section 3.1. The advantage of using this algorithm it can construct the multicast tree in short time comparing with heuristic algorithms that required a long time, moreover, we will build only the tree to the current receiver and will not build MST that required more time and is can't present an optimum solution to this problem.

- ***Group events Management module:*** Whenever controller receives join in or leave message from current multicast sessions this module is used for updating multicast tree and install new rule to the forwarders. In this module, we use Tabu Search algorithm to update the current multicast tree by generating neighbor's solutions of the current tree with the new receiver and choice the best solution for updating the multicast tree. We use the feature of SDN controller to calculate K-shortest path from source to every destination on offline mode and then using the pre-cached backup paths by TS algorithm to update multicast tree in short time so we can reduce the latency time required to join new members and find more optimum solution using this heuristic algorithm compared to Shortest Multicast tree. The details of this module are described in section 3.2.

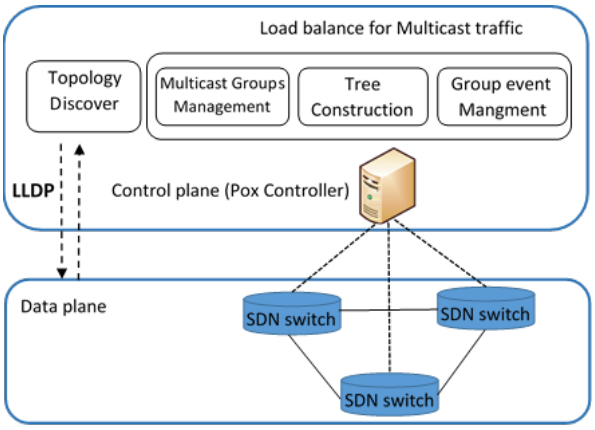

Fig. 1 Proposed Architecture

3. The Purposed Algorithms

In these subsections, we will describe in details the implementations of algorithms proposed in *tree construction module* and *group events Management module*. In our proposed scheme we will use three different algorithms 1- Dijkstra algorithm to construct shortest path tree. 2- Tabu Search algorithm to update multicast tree. 3- K-shortest paths algorithm [10] to find a K of paths between the source and each receiver in the network and caching these calculated paths in our controller so we can use it for fast update multicast tree based on Tabu Search algorithm, by this way we reduce the time required by TS to update multicast tree.

3.1 Multicast tree construction module

The main function of this module is to construct multicast tree when the controller needs to start a new multicast session. We use Dijkstra shortest path algorithm to construct the multicast tree. This algorithm used only one time to the initialize multicast tree and don't use for updating the multicast tree for the following reasons: 1- when the controller start to initialize multicast tree the number of the receivers almost small, so this algorithm can construct the tree in short time. 2- This algorithm construct shortest path tree that has the least cost from source to every receiver but it can't minimize the total number of links on the tree, so use it only in the first stage of multicasting (initialization). By this way, we take the advantage of this algorithm (i.e., fast constructing multicast tree) and avoid the shortage of this algorithm (i.e., can't minimize multicast tree). The details of this algorithm are shown in Fig. 2.

Dijkstra's Algorithm
Input: G = (V,E), Source and multicast group M Output: a tree cover all receivers in this group

1: T={S};d[S]←0; d[u]←∞ and pred[i]←nil for each u≠S, u€V
2: insert u with key d[u] into the priority queue Q, for each u€V
while (Q ≠ ∅)
3: j ← Extract-Min(Q)
4: for every node i, i €T and i is adjacent to j
alt = d[j] + ew(j, i) // ew: is edge weight =1for all links
if alt < d[i] then
d[i] ← alt
pred[i] ← j // set i as a child node of j
if node (i) not inclue in T, add the node into T
if all nodes in M join to T stop.
5: return T.

Fig. 2 Dijkstra for multicast tree

The main function of this module is to update multicast tree with the near optimum tree by minimizing the total number of links in the tree. We used tabu search algorithm [11] and K-shortest path algorithm with the global information available by the centralized controller (i.e., network information) to update multicast tree with near optimum solutions.

TS is a higher level heuristic procedure for solving the optimization problem, designed to guide other methods or their component processes to escape the trap of local optimality. TS has obtained optimal and near optimal solutions to a wide variety of classical and practical problems in applications ranging from scheduling to telecommunications and from character recognition to neural networks. It uses flexible structures memory (to permit search information to be exploited more thoroughly than by rigid memory systems or memory less systems), conditions for strategically constraining and freeing the search process embodied in tabu restrictions and aspiration criteria. Fig. 3 show TS algorithm flowchart for the optimization problem and Fig. 4 show our proposed procedure to update multicast tree the description of this algorithm as follow.

The algorithm first encourages the move to worth solution if there is no any improvement in the current solution. The tabu list introduced to discourage the search from coming back to previously-visited solutions and sure scape from local solution to global solutions. We used the following fitness function to evaluate each solution to select the best one for next iteration and final can find the near optimum solution.

$$F(T)= [C_T\ ]^{-1} = \left[\sum_{i,j \forall e_{ij} \in T} C_{ij} \right]^{-1} \quad (1)$$

Here CT is the summation of the total link cost in the multicast tree and Cij represents the cost of each link and it is a predefined value by the network administrator. In next section, we will give an example show how TS find near optimum multicast tree when we update multicast tree for any new join request.

First, the algorithm uses the current multicast tree as an initial solution. Then the algorithm used pre-cached backup list shown in table 1 to great more than one neighbor to the current multicast tree and select the best solution for next iteration. After joining the new destination to current tree. In each iteration, the TS randomly selected path from source to any destination and replaced this path by another backup path. If there is no farther improvement on the best solution then this solution will be used for updating multicast tree. Fig. 5. an example shows how this module works for updating multicast tree.

In this figure, TS algorithm generates three neighbor solutions to the current solution and the best one will be selected to join the new member to the current multicast session. The neighbors solution generated based on pre-defined backup paths. Then, for more optimization the algorithm will repeat a specified number of iterations in each iteration it randomly selects one destination and repeats the same method to generate neighbor solution using backup paths and select the best solution for next generation. In the case of leave member from the current multicast group, only the algorithm will burn it path from the tree.

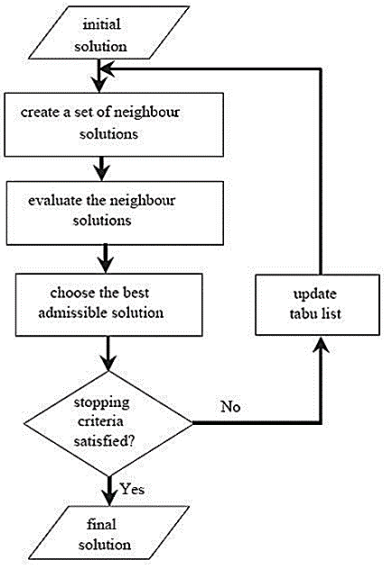

Fig. 3 Tabu Search flowchart

Procedure: management multicast events (leave and join)

X : current solution
M : The new distention that will be add to the current tree
K : Number of generated neighbors of X (current solution)
L (K) : backup list of paths from source to destinations
X' : Best neighbor solution
N(X) : Neighborhood of solution X
Fit : Fitness for solution X
Fit' : Fitness for solution X'

Begin
For I= 2 to K
Get from L (K) the backup list for destination M
Generate K of neighbor solution N(X) by attach the K backup path to current solution.
Get **Fit'** for each new neighbor solution.
X' = Find best solution in N(X)
Select X' as new multicast tree
End

Fig. 4 Steps of Tabu Search for updating multicast tree

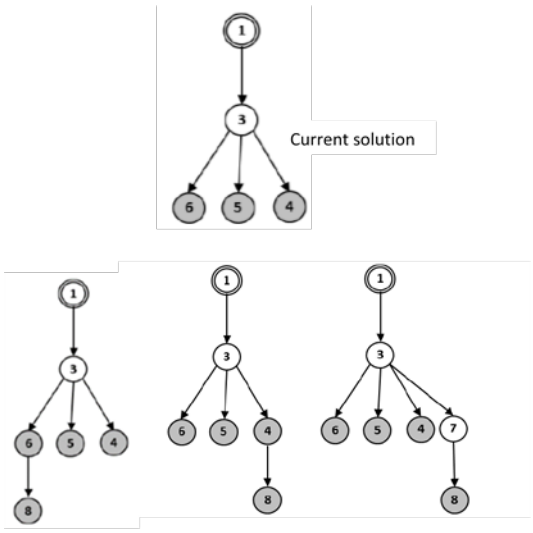

Fig. 5 Generating neighbor solutions from the current tree

4. Experimental Results

In this section, we test our method by comparing it with Dijkstra shortest path tree algorithm in [8]. Our method is implemented as OpenFlow controller modules, we use POX controller [9] and Mininet (12) to emulate the network. We used random connected topology generated using the Waxman generator provided by BRITE.

We assume that all link in the network have link capacity 20Mb/s. We use 720p video in a variable bit-rate MPEG4 format for the multicast session using (VLC application). A machine with core i3 processor and 8G of Ram are used for this emulation.

In Fig. [6] We tested the delay time required by the controller to construct the multicast tree at initialization of the multicast secession. We use different topology size of 20 nodes to 80 nodes. We use one host to be a source for video streaming (h0), and the number of receivers (i.e., group size) start by four receivers with the network of size 20 nodes and increased each time by 2 receivers with increasing the network size.

Fig. [6], shows that our methods can minimize the delay time required by the controller to start the multicast session compared to the other method. This because our method uses Dijkstra shortest path algorithm to construct the multicast tree from source to receivers only and then use Tabu Search algorithm to update the multicast tree when any new receiver want to join the multicast session. But in the other method purposed in [8], it constructs Minimum Spanning Tree that covers all switches in the network and then installs only the flows that represent the active receivers from the MST, so, it required more delay to construct MST.

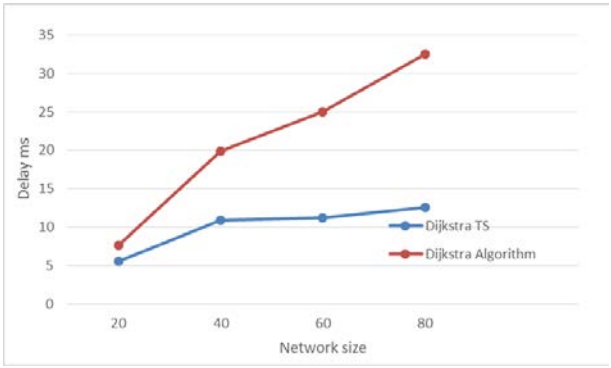

Fig. 6 Delay time for initializing multicast tree.

Table 2. Shows the installed flows in OpenFlow switches to forward the multicast data based on the constructed multicast tree. A large number of flows means the constructed multicast tree can't minimize the total hops in the tree.

In this emulation we start all multicast session with one source and 3 receivers, then 2 other receivers will request the controller to join the multicast session. In this case, the controller will update multicast tree to cover the new 2 receivers. Our method uses the backup paths to each destination and using TS algorithm will find the more optimum path to update each receiver. But, in the other method will use the back-up paths in the constructed MST to update the multicast tree.

We see from results in Table 2 that our method can reduce the total number of flows required to update multicast tree compared to another method. This means that our method is able to minimize the multicast tree with any update in the tree to join new receiver to multicast session.

The reason that make our algorithm able to minimize multicast tree is that, Tabu search is kind of heuristic algorithm that calculate Minimize Steiner Tree, and it is known that MST can minimize the total number of hops in the multicast tree, but it take a long time to construct multicast tree, so we use it to just update multicast tree based on pre-calculated back-up paths and used Dijkstra algorithm to initialize multicast tree because it take short time. But the other method use Dijkstra shortest path algorithm for both initialize and update multicast tree and it know that this algorithm can find the least cost of each path but can't minimize to total links in the multicast tree.

Table 2: Total number of flows installed per each method

	Group 1	*Group 2*	*Group 3*
Dijkstra	9 flows	10 flows	10 flows
Dijkstra TS	7 flows	6 flows	7 flows

Conclusion

In this paper, we present an efficient multicasting approach in SDN. For constructing multicast tree we purposed a new algorithm as a function in the controller, this algorithm takes advantage of Shortest Path Tree algorithms represented by Dijkstra algorithm and Minimum Sterner Tree algorithms represented by Tabu Search algorithm and avoid the shortages of both. Dijkstra algorithm can construct the multicast tree with short time but can't minimize the resulting multicast tree, in the other hand TS can minimize the resulting tree but it take a long time. By using Dijkstra to construct multicast tree at initialization of session and using TS to update multicast tree in case of joining any new receivers we get better performance compared to using the only shortest path tree algorithm. In this method, we use the good feature on SDN by using the centralized controller to construct the back-up paths for TS algorithm for fast update the multicast tree.

We evaluate our approach using Mininet network emulation with POX controller. Our evaluation indicates that our proposed multicasting mechanism is effective in delay time required to construct the multicast tree and it can optimize the tree with any update in multicast session.

References

[1] H. Holbrook, B. Cain, and B. Haberman. Aug. 2006 "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow, 2008 "enabling innovation in campus networks". ACM SIGCOMM Computer Communication Review, 38(2):69–74.

[3] C. Marcondes, T. Santos, A. Godoy, C. Viel, and C. Teixeira , Jul. 2012 "CastFlow: Clean-slate Multicast Approach using In-advance Path Processing in Programmable Networks", IEEE Symposium on Computers and Communications (ISCC).

[4] J. Moy, Mar. 1994 "Multicast Extensions to OSPF," RFC 1584 (Historic), Internet Engineering Task Force. [Online]. Available: http://www.ietf.org/rfc/rfc1584.txt.

[5] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, Jun. 1997 "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC 2117 (Experimental), Internet Engineering Task Force, obsoleted by RFC 2362. [Online]. Available: http://www.ietf.org/rfc/rfc2117.txt.

[6] K.-K YAP; T.-Y. HUANG; DODSON, B.; LAM, M.S.; MCKEOWN, N. 2010. Towards Software-Friendly Networks. In: ACM ASIA-PACIFIC WORKSHOP ON SYSTEMS, 1, New York, 2010. Proceedings… New York, p. 49-54.

[7] A. C. Cesar Teixeira, 2012 "CastFlow: Clean-slate multicast approach using in-advance path processing in programmable networks", ISCC, 2012, 2013 IEEE Symposium on Computers and Communications (ISCC), 2013 IEEE Symposium on Computers and Communications (ISCC) 2012, pp. 000094-000101, doi:10.1109/ISCC.6249274.

[8] J. -Ruey Jiang1, W. Yahya1,2, and M. Tri Ananta, 2011, "Load Balancing and Multicasting Using the Extended Dijkstra's Algorithm in Software Defined Networking" , © Springer-Verlag Berlin Heidelberg.

[9] M. Imase and B. M. Waxman, 1991, "Dynamic Steiner tree problem," SIAM lournal on Discrete Mathematics, vol. 4, no. 3, pp. 369-3S4.

[10] D. Eppstein. 1994. Finding the k shortest paths. 35th IEEE Symp. Foundations of Comp. Sci., Santa Fe, 1994, pp. 154-165. Tech. Rep. 94-26, ICS, UCI, 1994.

[11] F. Glover. 1989. "Tabu Search - PART 1". ORSA Journal on COMPUTING 1 (2): 190–206. doi:10.1287/ijoc.1.3.190.

[12] http://mininet.org/

Alaa Allakany received M.S. Degree in Computer Science Department, Faculty of Science from South Valley University, Egypt. And received B.S. in Faculty of science form South Valley University, Egypt. He is a Second year Ph.D. student and belongs to the department of Advanced Information Technology, Graduate school of Information Science and Electrical Engineering, Kyushu University, Japan.

Koji Okamura is a professor at Department of Advanced Information Technology and also at Computer Center, Kyushu University, Japan. He received B.S. and M.S. Degree in Computer Science and Communication Engineering and Ph.D. in Graduate School of Information Science and Electrical Engineering from Kyushu University, Japan in 1988, 1990, and 1998, respectively. He has been a researcher of MITSUBISHI Electronics Corporation, Japan for several years and has been a Research Associate at the Graduate School of Information Science, Nara institute of Science and Technology, Japan and Computer Center, Kobe University, Japan. He is Interested in Internet and Next Generation Internet, Multimedia Communication and Processing, Multicast/IPv6/Qos, Human Communication over Inherent and active Networks. He is a member of WIDE, ITRC, GENKAI, HIJK projects and Key person of Core University Program on Next Generation Internet between Japan and Korea sponsored by JSPS/KOSEF.