A Hybrid Routing Algorithm for an Efficient Shortest Path Decision in Network Routing

Taehwan Cho, Kyeongseob Kim, Wanoh Yoon and Sangbang Choi*

Department of Electronics Engineering, Inha University, Incheon, Korea

*burujo@naver.com, sangbang@inha.ac.kr**

Abstract

Recently, shortest path tree construction is essential in network routing. Dijkstra algorithm, one of the static routing algorithms, is widely used. When some links develop new weights, dynamic routing algorithms become more efficient than static routing algorithms. This is because dynamic routing algorithms reduce the redundancy caused by re-computing the affected part of the network in regards to the changed links. However, dynamic routing algorithms are not always efficient in some cases and increase the computation time when making the shortest path tree. In this paper, we present a Hybrid Shortest Path Tree (HSPT) algorithm which reduces the total execution time of shortest path tree computation by using the advantages of both static and dynamic routing algorithms. Comparisons with the other routing algorithms such as Dijkstra, Dynamic Dijkstra and RDSP show that the HSPT algorithm provides a better performance as demonstrated by the decrease in the execution time.

Keywords: *Hybrid routing, dynamic routing, Dijkstra, shortest path, network routing*

1. Introduction

Network Routing is a process which is choosing a way of sending communication data in a certain network. Network routing process is usually performed based on a routing table that manages various network destinations' routes. Therefore, the routing table formation written in a memory of the router is very important for effective routing. There are many graph algorithm methods used in routing algorithms. In graph G=(N,L), N stands for Nodes and L stands for a set of Nodes. Each link is composed of a pair of Nodes. In a viewpoint of network routing, the nodes of graphs represent routers, and the links which connect these nodes represent physical links between routers.

In today's Internet, demands for broadband Internet Applications have grown rapidly. Therefore, high speed routing has become more important at Open Shortest Path First (OSPF) which is the most used intra-autonomous system routing protocols. When topological changes occur due to an unexpected situation at the OSPF, network routing algorithms are used to update the routing table. For example, if there is a link failure in a network, then the Shortest Paths have to be re-computed. Normally in this case, the shortest paths computation is performed by re-running the Dijkstra algorithm [1, 2]. However, when links acquire new weights in a network, using the Dijkstra algorithm can increase the computation time and cause unnecessary corrections by repeating its operation in all of the nodes where the link's weight does not change. Therefore, this can cause network instability, since the overall routing table is frequently updated [3, 4].

This problem has motivated the introduction of dynamic routing algorithms used for the shortest path computation. Dynamic routing algorithms have been studied by Frigioni,

Narvaez, and others; the Dynamic Dijkstra [5] and RDSP (Reliable Dynamic Shortest Path) [6] methods are the result. These dynamic routing algorithms need more computation time for one node than the static routing algorithms. However, they can reduce their total computation time through the decrease in the number of nodes that have to be computed. That is, when some links have new weights in a network, Dynamic routing algorithms only find the affected nodes by the changed links when re-computing the shortest paths. Therefore they need less computation time to determine the shortest paths than that needed by the static routing algorithms. However, dynamic routing algorithms can require more computation time dependant on the location of the changed links.

In this paper, we present the Hybrid Shortest Path Tree (HSPT) algorithm, which uses less computation time by combining the advantages of the static routing algorithms and the dynamic routing algorithms. When evaluating the proposed algorithm, we compared it to the well-known static Dijkstra, the dynamic Dijkstra, and the RDSP algorithms.

2. Related Works

2.1. Dijkstra Algorithm

Dijkstra algorithm calculates the shortest path between a starting point and an ending point given in a direction graph which does not have negative link value. Therefore, it is a very effective method to find the shortest path from one point to the other point. If we apply this method to network, we can solve the problem of the shortest path from one point by supposing that a link's weight is not negative but weight direction graph G=(N,L). This algorithm which can find both the shortest path and the shortest distance is used in application areas to find a best way to the destination such as railway constructions, communication network design, and flight planning. Dijkstra algorithm is used a lot in network routing design; the prominent example is OSPF, a routing protocol in IP network.

The performance process of Dijkstra algorithm is like this. First, Dijkstra algorithm finds the closest point from a starting point. Then, the distance to the closest point is the shortest distance. We call the group of these closest points as S. Dijkstra algorithm keeps finding the closest point among points is not included in group S and this new closest point is one of the neighboring points of group S. The distance to the new closest point is the new shortest distance, and the new closest point now belongs to group S. Dijkstra algorithm repeats this process until it cannot find a new closest find, which means that it repeats until all points belong to group S.

2.2. Dynamic Routing Algorithm

Protocols in AS like OSPF mainly use Dijkstra algorithm. The core of Dijkstra algorithm is to find the shortest distance from a root node to every destination. However, Dijkstra algorithm calculates the shortest path from the very beginning without using the previous shortest path information in the routing tables if there is a small change in network topology. Accordingly, the shortest path calculation performed by Dijkstra algorithm can be a heavy burden for a router. Additionally, using entire CPU in this process is also problematic.

The routing table does not reflect actual network topology in a gap phase until it finishes the calculation. This is because it takes a very long time to calculate the shortest path by Dijkstra algorithm if there is a small change in network topology. In this case, if some important data packet moves on the route in a gap phase, loss of

packet occurs. This scenario can be happened in any time when there is some problem of the link in network, which is very a dangerous case.

Hence, if we are able to reduce this gap phase, we can prevent the packet loss. After all, how much we can reduce the gap phase is a matter of grave concern in network. For this reason, a dynamic algorithm should be preferred to a static algorithm since we can save more time for calculation by making full use of the shortest path information of previous status.

Another reason that we should use a dynamic algorithm is that 65% of the new shortest path after change is same as the previous shortest path even if there is some change in part of network topology. Even 35%, the rest of the new shortest path is very similar to the previous shortest path. Therefore, calculating the shortest path by a static algorithm is inefficient and calculating only the changed part of topology by a dynamic algorithm is efficient.

2.3. Problem Definition

The Dijkstra algorithm, one of the static routing algorithms, is widely used in network routing. However, this well-studied static routing algorithm is very ineffective when the link status changes in a network, which requires only the update of a small part of the old shortest paths. This is because it re-computes the new shortest paths in their entirety, not using the old shortest paths information. This is the hardest operation for a router; it occupies a great deal of CPU resources. It takes a very long time to compute the shortest paths using the Dijkstra algorithm; it also incurs a gap phase. In a gap phase, the routing table does not reflect the actual network topology until it finishes its computation, thereby packet losses can occur, which is problematic [7, 8].

For all of these reasons a dynamic routing algorithm should be preferred over a static routing algorithm, since it reduces the computation time by culling the majority of its information from the old routing table and only re-computing the nodes affected by updated shortest paths list.

However, a dynamic routing algorithm needs more computation time than a static algorithm, which depends on the location of the changed links. That is, when some links have new weights near the root node, it takes a long computation time being that a lot of nodes are affected by the updated shortest paths. Contrarily, near the end node, it takes a shorter computation time due to the few nodes affected. Therefore, dynamic routing algorithms do not get consistent results since they have wide variations for each different computation time case. This is problematic in network routing.

3. Proposed Algorithm

In this paper, we present a hybrid routing algorithm which reduces the total SPT execution time using the advantages gleaned from the static and dynamic routing algorithms. We call this algorithm the Hybrid Shortest Path Tree (HSPT). In order to efficiently compute the shortest paths using HSPT, the times when the static and dynamic algorithms are applied are very important.

Static routing algorithms should be applied when computing the shortest paths where some links have new weights near the root node. The reason that static routing algorithms are applied in this situation is because there are a lot of nodes which have to be computed near the root node. In this case, using static routing algorithms is a better method to compute the shortest paths rather than using the dynamic routing algorithms which need more computation time for each node.

```
Step 1 : Find the depth of whole network
Find the network depth by DFS(Depth First Search)
Determine D as 40% depth of whole network
for All nodes do
      Find the link whose link cost is changed
      Calculate the link depth by DFS
            if the link depth < D then
                  Go Step 2
            else
                  Go Step 3
            endif
endfor
Step 2 : Static routing
Find shortest paths by static routing
  G=(V,E) // using Dijkstra algorithm
Update the routing table
Step 3 : Dynamic routing
Find shortest paths by dynamic routing
  G=(V,E) // using Dijkstra algorithm
  Initialization des(e) are updated
  following the sequence of DFS from node e in SPT
    // all descendants of eupdated
Remove edges from Q which have end nodes belonging to des(e)
Update the old information in Q
Obtain a Temporary SPT
While(des(e))
  {des(e), mis_inc} ← extract(M)
If v has incoming links between des(e), then
  if D(i) from incoming link > D(j) from inner nodes,
     then D(i) = D(j)
  endif
endif
Update the routing table
```

Figure 1. Pseudo Code for the Proposed HSPT

Alternatively, dynamic routing algorithms should be applied to compute the shortest paths when some links have new weights near the end node. The reason that dynamic routing algorithms are applied in this case is that there are only a few nodes which have to be computed near the end node. In this case, using dynamic routing algorithms to re-compute only the nodes affected by old shortest paths is better than using the static routing algorithms which re-compute every node.

The hybrid routing algorithm is performed using the following procedures: First, in order to decide which algorithms should be applied, we need to find the depth of whole network using the Depth-First Search (DFS) method. Second, when some links have new weights of less depth than 40% of, static routing algorithms are applied to compute the shortest paths. Finally, when some of the links have new depth weights of more than 40% dynamic routing

algorithms are applied to compute the shortest paths. Figure 1 shows the pseudo code for the proposed HSPT.

The criterion for applying static and dynamic algorithms is a network depth of 40%, based on our simulations. In order to obtain the best criterion for applying the static and dynamic algorithms, we simulated them from depths of 30% to 60%. We found that 40% of the network depth was the best criterion for the hybrid routing algorithms. Figure 2 shows the simulation result of criterion.

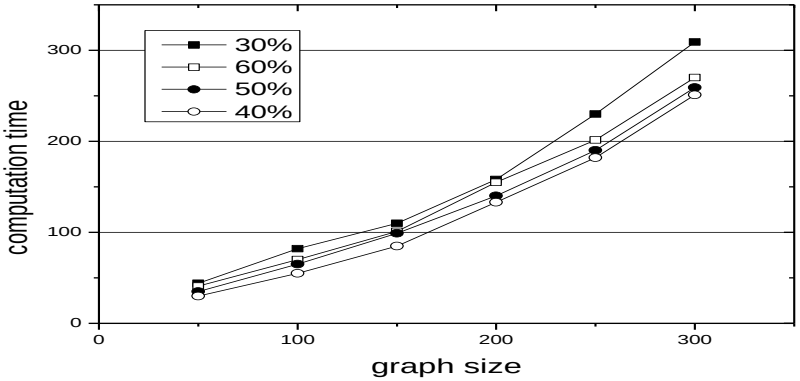

Figure 2. The Simulation Result of Criterion

4. Performance Evaluation

Table 1. The Input Parameters

Parameters	Values
Number of nodes	50, 100, 150, 200, 250, 300
Changed rate of link weights (%)	100, 200, 300, 400
Deviation of link weights	5, 10, 15, 20

The performance of the HSPT has been compared to previously published algorithms: the Dijkstra algorithm, Dynamic Dijkstra algorithm, and RDSP algorithm. The number of nodes, the changed rate of link weights, and the deviation of link weights were used for the input parameters in the simulations. The input parameters are restricted to the following values presented in Table 1.

4.1. Number of Nodes

The number of nodes represents the number of nodes in the entire network, i.e. the network size. Every simulation result was simulated in graphs which have 50 nodes, 100 nodes, 150 nodes, 200 nodes, 250 nodes and 300 nodes. Therefore, the result of a changed rate of link weights and a deviation of link weights were compared by simulating the graphs which have different numbers of nodes.

4.2. Changed Rate of Link Weights

The changed rate for the link weights is the rate used to compare it to the old link weight. In this paper, the case that the link weight was changed from 7 to 24 is shown. It is necessary to compare the performance depending on the altered link weight because it could have the different computation time for the shortest path by the altered link weight. We simulated this by changing the link weight to 100%, 200%, 300%, and 400%.

4.3. Deviation of Link Weights

The deviation of the link weights is the difference between the link weights and the mean value. In this study, the mean value is set to 10. At this time, we simulated by changing the deviation as 5, 10, 15, and 20.

4.4. Simulation Results

Comparisons are done based on the total execution time. The execution time is the computation of the shortest path in the given network. From Figure 2 to Figure 9 show the total execution time with respect to the number of nodes. The changed rate of link weights of Figure 2 is 100%, Figure 3 is 200%, Figure 4 is 300% and Figuer 5 is 400%. Also, the deviation of Figures 2, 3, 4 and 5 is 10 respectively. Next, the deviation of Figure 6 is 5, Figure 7 is 10, Figure 8 is 15 and Figure 9 is 20. Also, the changed rate of link weights of Figures 6, 7, 8 and 9 is 200% respectively. And the mean value of all figure is 10 respectively.

As shown in the all figure, HSPT algorithm outperforms Dijkstra algorithm, Dynamic Dijkstra algorithm and RDSP algorithm. As long as the number of nodes is increased, the computation time for shortest paths takes longer. If the changed rate of link weights is low, then the computation time for shortest paths becomes longer linearly. However, If the changed rate of link weights is high, then the computation time for shortest paths becomes longer nonlinearly.

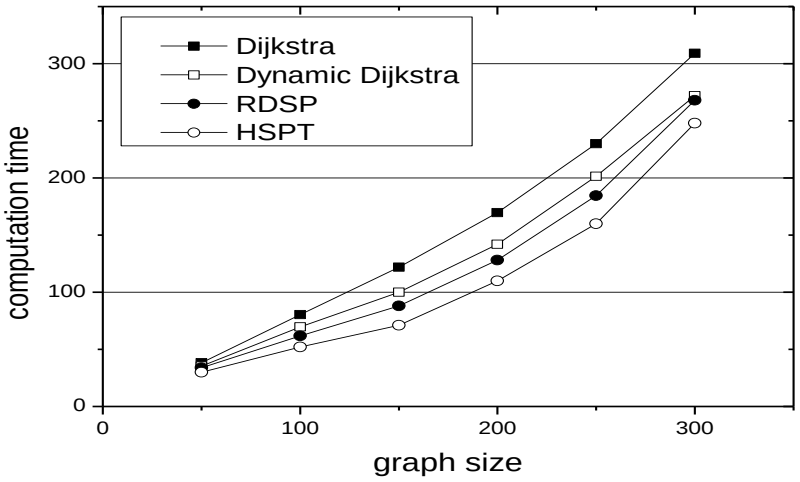

Figure 2. Total Computation Time with Respect to the Number of Nodes and the Changed Rate of Link Weights 100

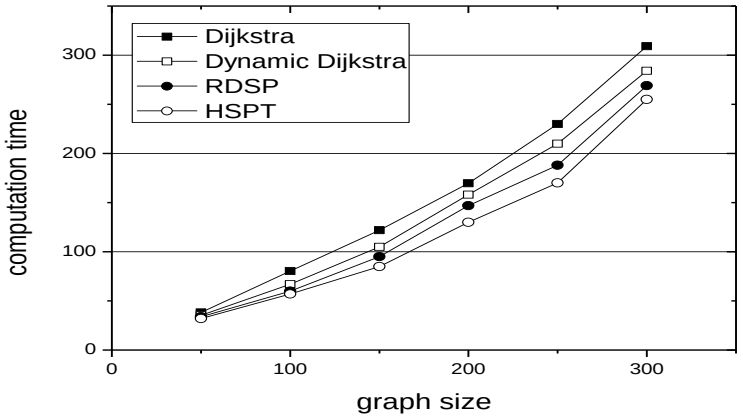

Figure 3. Total Computation Time with Respect to the Number of Nodes and the Changed Rate of Link Eeights 200

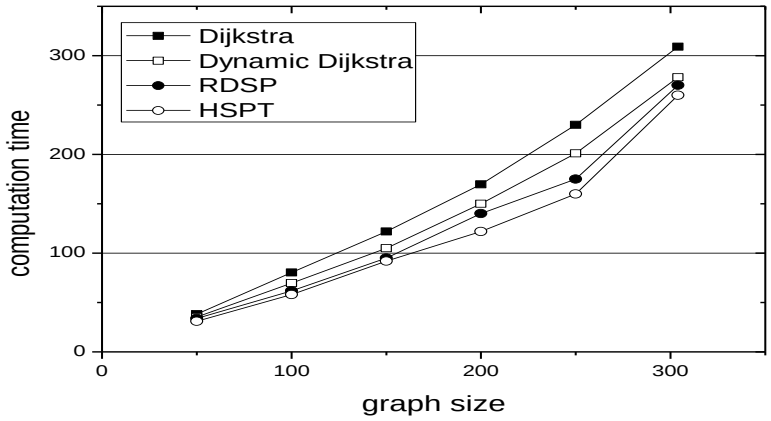

Figure 4. Total Computation Time with Respect to the Number of Nodes and the Changed Rate of Link Weights 300

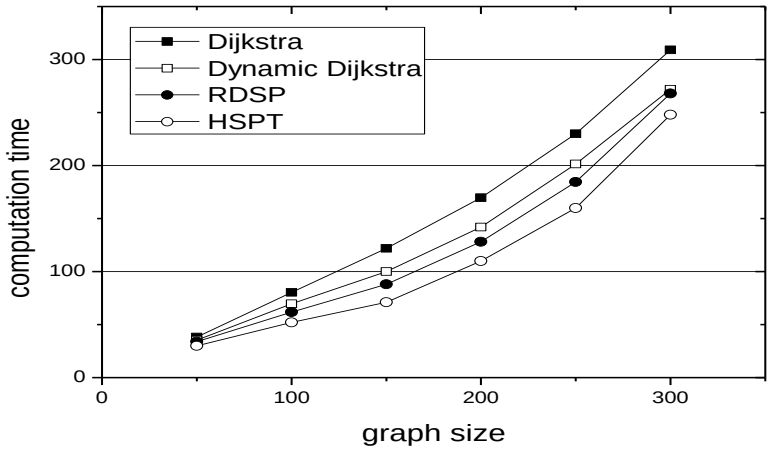

Figure 5. Total Computation Time with Respect to the Number of Nodes and the Changed Rate of Link Weights 400

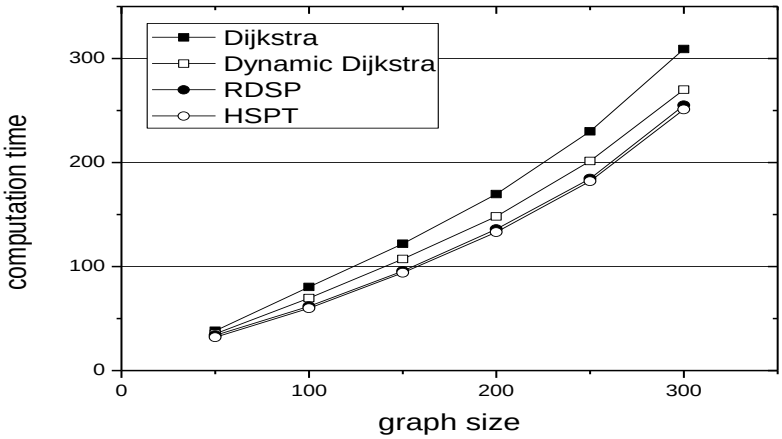

Figure 6. Total Computation Time with Respect to the Number of Nodes and the Deviation of the Link Weights 5

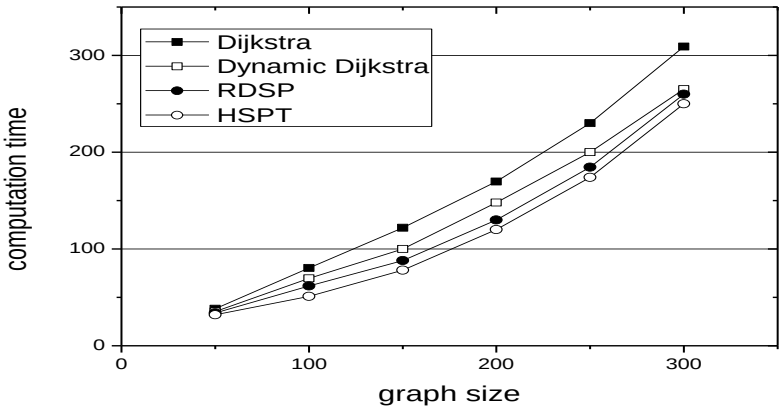

Figure 7. Total Computation Time with Respect to the Number of Nodes and the Deviation of the Link Weights 10

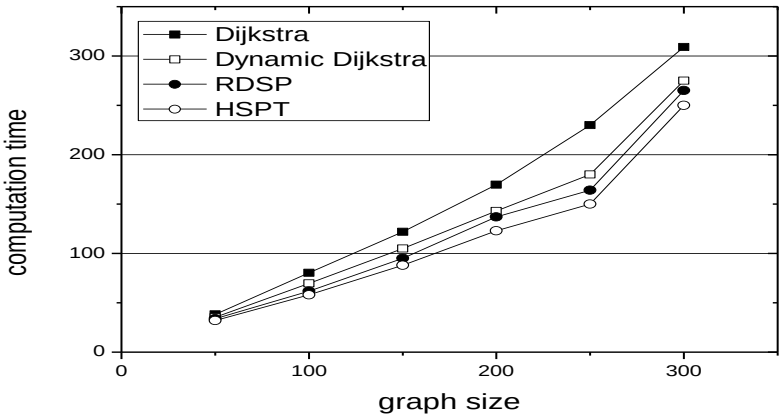

Figure 8. Total Computation Time with Respect to the Number of Nodes and the Deviation of the Link Weights 15

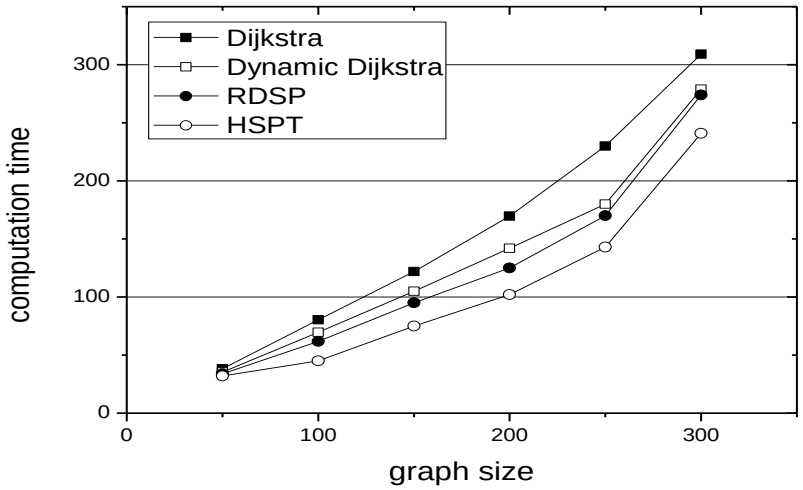

Figure 9. Total Computation Time with Respect to the Number of Nodes and the Deviation of the Link Weights 20

5. Conclusion

In this paper, we present the HSPT (Hybrid Shortest Path Tree) algorithm in order to offer an efficient shortest paths decision used to reduce the total execution time by using the advantages of the static and dynamic algorithms. Less total execution time leads to reduction in packet loss. As shown in the comparison results, the proposed HSPT algorithm provides a better performance when compared to the Dijkstra, Dynamic Dijkstra, and RDSP methods in terms of the computation time of the shortest path.

Acknowledgements

This work was supported by Key Research Institute Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012-0005858).

References

[1] E. Dijkstra, "A note two problems in connection with graphs", Numerical Math, vol. 1, **(1959)**, pp. 269-271.
[2] G. S. Cho and J. K. Ryeu, "An Efficient Method to Find a Shortest Path for a Car-Like Robot", International Journal of Multimedia and Ubiquitous Engineering, vol. 1, **(2006)**, pp. 1-6.
[3] B. Xiao, J. Cao, Z. Shao and E. H. M. Sha, "An Efficient Algorithm for Dynamic Shortest Path Tree Update in Network Routing", Journal of Communication and Networks, vol. 9, **(2007)**, pp. 499-510.
[4] S. Y. Ameen, A. A. Ahmed and I. A. Ibrahimi, "MANET Routing Protocols Performance Evaluation with TCP Taho, Reno and New-Reno", International Journal of u - and e - Service, Science and Technology, vol. 4, **(2011)**, pp. 37-49.
[5] E. P. F. Chan and Y. Yang, "Shortest Path Tree Computation in Dynamic Graphs", IEEE Transactions on Computers, vol. 58, **(2009)**, pp. 541-557.
[6] T. H. Cho, J. W. Kim, B. J. Kim, W. O. Yoon and S. B. Choi, "A Study on Shortest Path Decision Algorithm for Improving the Reliability of Dynamic Routing Algorithm", Journal of the Korean Institute of Information Scientists and Engineers, vol. 38, **(2011)**, pp. 450-459.
[7] V. Eramo, M. Listanti and A. Cianfrani, "Design and Evaluation of a New Multi-Path Incremental Routing Algorithm on Software Routers", IEEE Transactions on Network and Service Management, vol. 5, **(2008)**, pp. 188-203.
[8] S. K. Lee, J. W. Jang, S. J. Jang and J. Y. Shin, "Development and Performance Analysis of ABR-DBA Algorithm for Improve Network Performance", International Journal of Future Generation Communication and Networking, vol. 1, **(2008)**, pp. 1-6.

Authors

Taehwan Cho received the B.S. degree in aerospace engineering from Inha University, Incheon, Korea, in 2001. He is currently working toward the Ph.D. degree in the computer architecture and networks Laboratory. His research interests include communication and security in computer networks, wireless mobile ad hoc and sensor networks.

Kyeongseob Kim received the B.S. degree in information communication engineering from Hannam University, Daejeon, Korea, in 2002, the M.S. degree in electronics engineering from Inha University. He is currently working toward the Ph.D. degree in the computer architecture and networks Laboratory. His research activities include computer architecture, computer networks, and wireless mobile ad hoc sensor networks and vehicle network system.

Wanoh Yoon received the B.S. degree in electronics engineering from Kyonggi University, GyeongGi-Do, Korea, in 2000, the M.S. degree and Ph. D. degree in electronics engineering from Inha University, in 2002 and 2010, respectively. He is currently working as a research professor in Inha University, since 2010.

His research activities include distributed and parallel system, computer architecture and ADS-B system.

Sangbang Choi earned the M.S. and Ph.D. in electrical engineering from the University of Washington, Seattle, in 1988 and 1990, respectively. He is currently a professor of electronic engineering at Inha University, Incheon, Korea. His research interests include computer architecture, computer networks, wireless communication, parallel and distributed systems.