K-Shortest Path 알고리즘에 기초한 새로운 대역폭 보장 라우팅 알고리즘

정회원 이 준 호*, 이 성 호**

New Bandwidth Guaranteed Routing Algorithms based on K-Shortest Path Algorithm

Jun-Ho Lee*, Seong-Ho Lee** *Regular Members*

요 약

본 논문에서는 MPLS 네트워크에서 LSP 설정에 적용될 수 있는 새로운 대역폭 보장 온라인 라우팅 알고리즘들을 제안하고 기존의 알고리즘들과 함께 그 성능을 시뮬레이션을 통해서 평가한다. 제안된 방식은 기존의 WSP나 SWP 알고리즘을 K-shortest loopless path 알고리즘에 기초해서 확장시킨 형태를 가진다. 시뮬레이션을 통해서 accepted bandwidth, accepted request number 그리고 average path length라는 성능을 평가한 결과, 모든 노드들이 LSP 설정의 ingress나 egress 노드가 될 수 있는 상황에서 제안된 방식들이 전반적으로 우수한 성능을 보였는데 네트워크 부하가 큰 경우에는 특히, 최소 홉 경로에 기초한 방식들이 좋은 성능을 보임을 알 수 있다.

Key Words : Routing algorithm, MPLS, k-shortest path

ABSTRACT

In this paper, new on-line routing algorithms with a bandwidth constraint are proposed. The proposed algorithms may be used for a dynamic LSP setup in MPLS network. We extend the WSP algorithm, the SWP algorithm and a utilization-based routing algorithm into the proposed algorithms by slightly modified K-shortest loopless path algorithms. The performances such as accepted bandwidth, accepted request number and average path length of the proposed and the previous algorithms are evaluated through extensive simulations. All simulations are conducted under the condition that any node can be an ingress or egress node for a LSP setup. The simulation results show that the proposed algorithms have the good performances in most cases in comparison to the previous algorithms. Under the heavy load condition, the algorithms based on the minimum hop path perform better than any other algorithms.

I. 서 론

최근 들어서 고속 인터넷 사용자들의 수가 증가함에 따라서 인터넷을 통한 고속의 데이터 전송 및 QoS(Quality of Service) 보장이 중요한 문제가 되고 있다. 그런 문제의 해결책으로 Multiprotocol Label Switching(MPLS)[1] 기술이 고려되고 있다. MPLS 네트워크에서는 ingress 노드에서 egress 노드까지 일정한 대역폭을 보장할 수 있는 Label Switched Path(LSP)를 설정할 수 있다. 본 논문에서는 그와 같은 LSP 설정에 적용될 수 있는 온라

* 서울산업대학교 전자정보공학과 (ljh@snut.ac.kr), **서울산업대학교 전자정보공학과 (shlee@snut.ac.kr)
논문번호 : 030176-0428, 접수일자 : 2003년 4월 28일

인 라우팅 알고리즘을 제안하고 기존의 알고리즘들과 함께 그 성능을 시뮬레이션을 통해서 평가해 본다.대역폭 보장은 대역폭을 제한 조건(constraint)로 하는 constraint-based QoS-Routing을 의미한다. 대역폭 외에도 end-to-end delay, jitter 혹은 loss 등이 QoS 제한 조건으로 지정될 수 있다. 하지만, 둘 이상의 제한 조건들을 갖는 문제는 NP-hard로 아직까지 실용적인 알고리즘이 제시되고 있지는 않다. End-to-end delay, jitter 혹은 loss 등의 파라메터는 링크의 대역폭에 비해서 정확한 값을 얻기가 어렵기 때문에 각 QoS 제한 조건을 둥가 대역폭(equivalent bandwidth)으로 변환해서 대역폭 보장 라우팅 알고리즘을 적용하는 방식이 더 실용적이라고 할 수 있다. 따라서, 실용적인 QoS 라우팅 알고리즘의 구현 측면에서 본다면 현재까지의 기술 수준에서는 효율적인 대역폭 보장 라우팅 알고리즘의 제시도 의미를 갖는다고 할 수 있다.

본 논문에서 다룰 라우팅 문제를 좀더 구체적으로 정의하면 다음과 같다. MPLS 네트워크(ATM 네트워크와 같이 임의의 대역폭을 갖는 경로를 설정할 필요가 있는 모든 네트워크)에서 임의의 ingress-egress 노드 쌍과 임의의 요구 대역폭을 갖는 LSP setup 요청이 각 노드로 독립적으로 하나씩 도착하는데 미래의 도착 정보에 대해서는 알 수가 없다. 이런 상황에서 네트워크 자원을 효율적으로 사용하면서 요구된 대역폭을 보장해주는 경로를 찾는 문제라고 할 수 있다. 그와 같은 문제를 다루는 기존의 라우팅 알고리즘들 중에서 본 논문에서 성능 평가에 사용한 대표적인 대역폭 보장 라우팅 알고리즘들을 살펴보면 다음과 같다.

- Widest Shortest Path(WSP)[2]: 최단 경로(shortest path)들이 둘 이상 존재할 경우 경로 대역폭(path bandwidth)(경로를 구성하는 링크들의 residual capacity 중에서 가장 작은 값)이 가장 큰 것을 선택하는 방식으로 DIJKSTRA 알고리즘[8]을 확장해서 구할 수 있다.

기본 개념은 DIJKSTRA 알고리즘을 수행하면서 다음에 marking 할 노드를 고를 때 최소 거리를 갖는 노드를 고르게 되는데 만일 그런 노드들이 여러 개 있는 경우에는 경로 대역폭이 가장 큰 노드를 택하는 것이다.

- Shortest Widest Path(SWP)[3]: widest path(경로 대역폭이 가장 큰 경로)들이 여러 개 존재할 경우 그 중 최소 거리를 갖는 경로를 선택하는 방식으로 기본적인 구현은 다음과 같이 한다. 먼저, widest path를 구한다. 그 widest path의 경로 대역폭이 B라고 하자. 네트워크에서 residual capacity가 B 미만인 모든 링크들을 제거한 후에 DIJKSTRA 알고리즘을 적용해서 최단 경로를 구하면 그것이 shortest widest path가 된다. Widest path를 구하는 것은 거리 함수(distance function)으로 경로 대역폭을 사용해서 DIJKSTRA 알고리즘을 적용하면 된다.

- Minimum Interference Routing Algorithm(MIRA)[4]: 어떤 송/수신 노드 쌍에 대한 경로를 설정할 때 다른 송/수신 노드 쌍들의 미래의 경로 설정에 최소한의 영향(minimum interference)을 미칠 수 있는 경로를 찾자는 것이다.

그림 1에서 각 링크의 residual capacity가 1이라고 가정하고 S3 -> D3 경로를 설정할 때 최소 홉 경로를 사용한다면 1->7->8->5 경로가 선택되는데 그 후에는 S1 -> D1 과 S2 -> D2 사이의 경로 설정이 불가능해 진다. 그러나, 최소 홉 경로가 아닌 1->2->3->4->5 경로를 선택하면 그와 같은 문제가 없게 된다. 이때, 1->2->3->4->5 경로가 1->7->8->5 경로 보다 interference가 작은 경로가 된다. 그러나, 이와 같은 MIRA algorithm은 네트워크의 모든 노드들이 송/수신 노드가 될 수 있는 환경에서는 - 실제 네트워크에서는 이와 같은 가정이 더 현실적이다 - 그 계산의 복잡성에 비해서 성능의 향상이 그리 크지 않게 된다.

그 이유는, 그림 1에서 이제 모든 노드가 송/수신 노드가 될 수 있다고 가정해보자. 그러면, 1->2->3->4->5 경로가 1->7->8->5 경로 보다 interference가 작은 경로가 된다는 보장을 할 수가 없게 된다. 그것은 1->2->3->4->5 경로도 1->2, 1->3, 1->4, 1->5, 2->3 등등의 경로 설정에 영향을 미치기 때문이다. 이런 경우 두 경로 사이의 interference 값의 차이가 그리 크지 않기 때문에 [4]에서 제시된 heuristic 알고리즘으로는 정확히 minimum interference를 구할 수 없고 따라서 [4]의 알고리즘으로 선택된 경로가 계산이 단순한 최소 홉 경로보다 더 효과가 좋다는 보장이 없게 된다. 오히려, 실제 interference를 더 크게 하는 경로를 선택할 수도 있게 된다.

본 논문에서는 [4]에서 제시한 2개의 알고리즘 WSUM-MAX 알고리즘(WSUMMIRA 알고리즘)과 LEX-MAX 알고리즘(LEXMIRA 알고리즘)의 성능

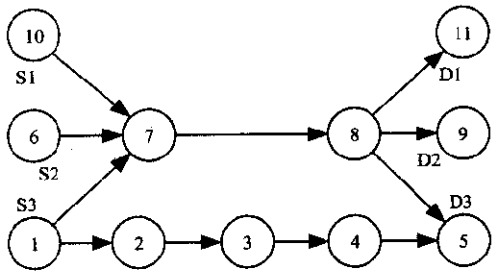


그림 1. MIRA 알고리즘의 기본개념
Fig 1. basic concept of the MIRA algorithm

을 평가한다. 또한, MIRA 알고리즘의 단점인 계산의 복잡성을 개선하기 위해서 제시된 방식들 중에서 [5]에서 제시된 dynamic online routing algorithm (DORA 알고리즘)과 [6]에서 제시된 2개의 알고리즘 Minimum Interference bottleneck link avoidance algorithm (MIBLASMIRA 알고리즘)과 Minimum Interference Path Avoidance algorithm (MIPASMIRA 알고리즘)의 성능을 평가한다.

본 논문이 대역폭 보장 온라인 라우팅 알고리즘 분야에 기여하는 바는 다음과 같다. 첫째로 현재까지의 대표적인 heuristic 알고리즘들을 넓은 범위의 부하 조건에서 시뮬레이션을 통해서 성능을 비교하는 것이다. 기존의 방식들이 대부분의 경우에 작은 범위의 부하에서 시뮬레이션을 한 결과를 토대로 제안된 방식들의 성능이 우수하다는 주장을 하고 있지만 heuristic 알고리즘의 한계로 인해서 넓은 범위의 부하에 대해서 과연 최적인 단일의 알고리즘이 존재하느냐가 의문이다. 만일, 그와 같은 단일의 알고리즘이 존재하지 않는다면 실제 MPLS 망에 어떤 알고리즘을 적용할 것인가가 문제가 될 것이다. 특정한 부하 상태에 따라서 알고리즘을 달리 적용하는 것은 비현실적이므로 본 논문에서는 가능하다면 실제 하나의 알고리즘을 선택하기 위한 기준을 제시하고자 한다.

둘째로는 앞에서도 언급하였지만 성능이 좋다고 알려진 MIRA 알고리즘이 과연 모든 노드들이 ingress-egress 노드가 될 수 있는 환경에서도 계산의 복잡성을 극복할 만한 성능의 향상이 있을 것인가 하는 의문을 해결하는 것이다.

본 논문의 구성은 다음과 같다. 2장에서 본 논문에서 제안하는 4가지 대역폭 보장 라우팅 알고리즘들에 대해서 설명하고 3장에서는 제안된 알고리즘들과 기존의 알고리즘들의 성능을 시뮬레이션을 통해서 평가해 본다. 마지막으로 4장에서 결론과 추후 과제를 보인다.

II. 제안된 대역폭 보장 라우팅 알고리즘

본 논문에서는 K-shortest loopless path 알고리즘 [7]에 기초한 4가지 대역폭 보장 라우팅 알고리즘을 제안한다. 본 논문에서 사용된 K-shortest loopless path 알고리즘은 기존의 방식이 K개의 shortest path들을 구하는데 비해서 본 논문에서는 K번째 path cost를 갖는 모든 경로들을 구하도록 수정하였다. 또한, K-shortest loopless path 알고리즘을 수정한 K-widest loopless path 알고리즘을 함께 사용한다.

먼저, 알고리즘 설명에 사용될 기호들을 정의하면 다음과 같다. G(N,L)은 네트워크를 나타내는 direct graph이고 $N = \{n_1,\ldots,n_M\}$은 노드 집합, $L = \{l_1,\ldots,l_V\}$는 링크 집합이며 M과 V는 각각 노드 수와 링크 수를 나타낸다. 노드 i에서 노드 j로의 링크는 (i,j) 혹은 l_{ij}로도 표시하며 일반적으로 $l_{ij} \neq l_{ji}$. $R = \{r_{ij}\}$는 각 링크의 residual capacity 집합이고 $C = \{c_{ij}\}$는 각 링크의 capacity 집합이다. $W = \{w_{ij}\}$는 각 링크의 weight 집합으로 w_{ij}는 링크 l_{ij}의 weight 값을 나타내며 각 링크의 delay 값이나 utilization 값으로 설정될 수 있다. 또한, 모든 링크들에 대하여 $w_{ij} = 1$로 하면 minimum hop loopless path를 구하게 된다. 노드 s에서 노드 d로의 shortest loopless path는 p(s,d)로 표현하며 알고리즘 수행에 따라서 k번째 계산된 것은 $p_k(s,d)$와 같이 표시한다. $M(p_k(s,d))$는 $p_k(s,d)$에 포함된 모든 노드들의 개수를 나타내며 $N(p_k(s,d))$는 $p_k(s,d)$에 포함된 모든 노드들의 집합, $A(p_k(s,d))$는 $p_k(s,d)$에 포함된 모든 링크들의 집합을 각각 나타낸다. Path $p_k(s,d)$에는 다음과 같은 두 가지 방식으로 경로 길이(path length)를 할당한다.

$L1(p_k(s,d)) = \sum w_{ij}$, $\forall(i,j)\in A(p_k(s,d))$: additive path length

$L2(p_k(s,d)) = \min\{w_{ij}\}$, $\forall(i,j)\in A(pk(s,d))$: bottleneck path length

비슷하게 path $p_k(s,d)$의 경로 대역폭 $B(p_k(s,d))$를 다음과 같이 정의한다.

$B(p_k(s,d)) = \min\{r_{ij}\}$, $\forall(i,j)\in A(p_k(s,d))$

경로 $p_k(s,d)$의 deviation 노드는 $d(p_k(s,d))$로 표현되며 경로 $sub_{pk(s,d)}(a,b)$는 경로 $p_k(s,d)$에 포함되며 경로 $p_k(s,d)$와 노드 a에서 노드 b까지 일치하는 경로를 나타낸다. 어느 한 경로의 시작 노드와 다른 경로의 마지막 노드가 동일한 경우 두 경로를 연결한 새로운 경로를 정의할 수 있는데 다음과 같이 표현한다.

$p(s,d) = p(s,a) + p(a,d)$

이것은 경로 p(s,a)의 마지막 노드(node a)와 경로 p(a,d)의 시작 노드(node a)가 동일한 경우로 p(s,a) + p(a,d) 는 p(s,a)의 시작 노드 s에서 p(a,d)의 마지막 노드 d까지의 새로운 경로 p(s,d)가 노드 s에서 노드 a까지는 경로 p(s,a)와 같고 노드 a에서 노드 d까지는 경로 p(a,d)와 같게 구성된다는 것을 의미한다. 경로 p(s,d-1)은 경로 p(s,d)에 포함되며 경로 p(s,d)와 노드 s에서 노드 d 바로 전의 노드까지 동일한 경로를 의미하며 경로 p_k에 속한 i번째 노드는 n^k_i로 표시한다.

이상의 기호들을 사용해서 제안된 알고리즘을 설명하면 다음과 같다.

1. KShortestPath Algorithm (K-shortest loopless path Algorithm)

Input: G(N,L), source node s, destination node d, link weight W, K
Output: shortest path들의 list KShortestPaths
Algorithm:
try to find the first shortest path $p_1(s,d)$ by DIJKSTRA algorithm. If there is no path, set KShortestPaths = null and return
set ShortestPathLength = $L1(p_1(s,d))$
insert the $p_1(s,d)$ into the priority queue CandidatePaths
set PathLevel = 1
set k = 1
while(the CandidatePaths is not empty)
{
  Remove the first path element p(s,d) from the CandidatePaths and set $p_k(s,d) = p(s,d)$
  if($L1(pk(s,d)) >$ ShortestPathLength)
  {// the current path has a greater path length
   // than the previous one
    set ShortestPathLength = $L1(p_k(s,d))$
    increment PathLevel by 1
    if(PathLevel > K)
      return // we found all the Kth shortest
             // loopless paths
  }
  insert the $p_k(s,d)$ into the output list KShortestPaths
  all nodes in $sub_{pk(s,d)}(s,d(p_k)-1)$ are removed temporarily
  remove arcs starting in $d(p_k)$ which belong to the other shortest paths already determined
  for(all nodes n^k_i in $sub_{pk(s,d)}(d(p_k),d-1)$)
  { remove arc (n^k_i,n^k_{i+1})
     try to find a shortest path $p(n^k_i,d)$ by DIJKSTRA algorithm
     if(found no shortest path) continue with next n^k_i in for loop
     // we found another shortest path
     set p(s,d) = $sub_{pk(s,d)}(s,n^k_i)$ + $p(n^k_i,d)$
     insert the p(s,d) into the CandidatePaths
     remove n^k_i
  }
  restore all the removed nodes and edges
  set k = k + 1
}

2. KWidestPath Algorithm (K-Widest loopless path Algorithm)

Input: G(N,L), source node s, destination node d, link residual capacity R, K
Output: widest path들의 list KWidestPathsAlgorithm:
K-Widest loopless path 알고리즘은 경로 길이 계산을 link residual capacity r_{ij}를 사용한 bottleneck path length $L2(p_k(s,d))$로 하고 두 노드 사이의 경로를 구할 때는 widest path 알고리즘을 사용한다는 점 외에는 K-shortest loopless path 알고리즘과 동일하다.

이상의 두 알고리즘을 기초로 해서 본 논문에서는 Widest K-Min-Hop(WKMH) path, Shortest K-Widest(SKW) path, Min Utilization K-Min-Hop(MUKMH) path, Random K-Min-Utilization-Hop(RKMUH) path라는 4가지 알고리즘을 제안한다.

1. WidestKMinHopPaths Algorithm (WKMH path Algorithm)

Input: G(N,L), source node s, destination node d, link weight W, link residual capacity R, K, requested bandwidth B

Output: B이상의 path bandwidth를 갖는 s에서 d까지의 path

Algorithm:

remove all links (i,j) if r_{ij} < B

set w_{ij} = 1, $\forall(i,j)\in L$

get a shortest path list by KShortestPath algorithm with the parameter K

find a path which has a maximum path bandwidth in the shortest path list

restore the removed links

if a path is found, update the link residual capacity along the path

WKMH path 알고리즘에서 link weight 값을 링크의 delay로 설정하면 최소 홉 대신에 최소 end-to-end delay를 갖는 경로들 중에서 그 경로 대역폭이 가장 큰 것을 찾게 된다. 그런 경로가 여러 개 있는 경우에는 임의로 선택된다. WKMH path 알고리즘은 기존의 WSP 알고리즘을 확장시킨 것으로 K번째 크기의 최단 경로들 중에서도 최종 경로를 선택함으로써 local search로 인한 단점 - 현재 상태에서는 최적이지만 미래의 상태에서는 그렇게 되지 못하는 것을 극복하기 위한 것이다. 그러나, 모두 heuristic 알고리즘이기 때문에 항상 WKMH path 알고리즘이 WSP 알고리즘에 비해서 성능이 좋아지는 것은 아니다. 다만, heuristic 하게 그럴 가능성이 커지도록 한 것이다.

2. ShortestKWidestPaths Algorithm (SKW path Algorithm)

Input: G(N,L), source node s, destination node d, link weight W, link residual capacity R, K, requested bandwidth B

Output: B이상의 path bandwidth를 갖는 s에서 d까지의 path

Algorithm:

remove all links (i,j) if r_{ij} < B

get a widest path list by KWidestPath algorithm with the parameter K

set w_{ij} = 1, $\forall(i,j)\in L$

find a path which has a minimum hop in the widest path list with respect to w_{ij}

restore the removed links

if a path is found, update the link residual capacity along the path

SKW path 알고리즘은 SWP 알고리즘을 확장시킨 것으로 기본 개념은 WKMH path 알고리즘과 유사하다.

3. MinUtilizationKMinHopPaths Algorithm (MUKMH path Algorithm)

Input: G(N,L), source node s, destination node d, link weight W, link residual capacity R, link capacity C, K, requested bandwidth

BOutput: B이상의 path bandwidth를 갖는 s에서 d까지의 path

Algorithm:

remove all links (i,j) if r_{ij} < B

set w_{ij} = 1, $\forall(i,j)\in L$

get a shortest path list by KShortestPath algorithm with the parameter K

find a path which has a minimum path utilization in the shortest path list

restore the removed links

if a path is found, update the link residual capacity along the path

MUKMH path 알고리즘은 K번째까지의 최단 경로들 중에서 path utilization 값이 가장 작은 것을 선택하는 방식으로 경로 $p_k(s,d)$의 path utilization $U(p_k(s,d))$는 다음과 같이 정의한다.

$$U(p_k(s,d)) = \sum\{(c_{ij} - r_{ij})/c_{ij}\},\ \forall(i,j)\in A(p_k(s,d))$$

4. RandomKMinUtilizationHopPaths Algorithm (RKMUH path Algorithm)

Input: G(N,L), source node s, destination node d, link weight W, link residual capacity R, link capacity C, K, requested bandwidth B

Output: B이상의 path bandwidth를 갖는 s에서 d까지의 path

Algorithm:

remove all links (i,j) if r_{ij} < B

set $w_{ij} = (c_{ij} - r_{ij})/c_{ij}$, $\forall(i,j)\in L$

get a shortest path list by KShortestPath algorithm with the parameter K
randomly choose a path from the shortest path list
restore the removed links
if a path is found, update the link residual capacity along the path

RKMUH path 알고리즘은 경로 길이를 link utilization의 합으로 계산하는 것으로 모든 링크들의 utilization이 동일하다면 RKMUH path 알고리즘은 최소 홉 경로를 찾는 것이 된다. RKMUH path 알고리즘은 heuristic 하게 홉 수가 작으면서도 link utilization이 작은 링크들로 구성되는 경로를 찾고자 하는 것이다.

Ⅲ. 성능 평가

이번 장에서는 기존의 대역폭 보장 알고리즘들과 본 논문에서 제안된 대역폭 보장 알고리즘들의 성능을 시뮬레이션을 통해서 평가해본 결과에 대해서 설명한다.

성능평가의 대상으로 irregular topology 2개와 1개의 regular topology 등 모두 3개의 네트워크를 사용하였다. 그림 2의 네트워크[6]는 MIRA 알고리즘의 성능 평가에 사용된 것으로 각 링크는 실제로는 두 개의 서로 다른 방향을 갖는 링크를 나타내며 굵은 표시가 된 것은 다른 일반적인 링크들에 비해서 link capacity가 큰 링크들을 나타낸다. 그림 3은 미국의 NSFnet T1 backbone 네트워크와 유사한 구조를 갖는 예이고 그림 4는 regular topology의 예로 4x4 mesh 네트워크를 나타낸다.모든 네트워크 환경에 대해서 모든 노드들이 LSP의 ingress 노드와 egress 노드가 될 수 있다고 가정하였으며 그림 2의 네트워크에서는 노드 수가 15이므로 전체 ingress-egress 쌍의 개수는 15x14 = 210개가 되며 그림 3과 4의 네트워크에서는 노드 수가 16이므로 전체 ingress-egress 쌍의 개수는 16x15 = 240개가 된다.

시뮬레이션은 동적인 트래픽 환경을 고려하였으며 각 노드로 평균값이 λ인 Poisson process에 따라서 LSP setup 요구가 도착한다고 가정하였다. LSP setup 요구의 egress 노드는 요구가 들어온 노드를 제외한 다른 노드들 중에서 동일한 확률로 하나가 선택된다. 요구된 LSP setup의 대역폭은 1,2,3 중에서 동일한 확률로 선택된다. 라우팅에 성공한 LSP setup이 네트워크에 머무는 시간 - LSP holding time은 평균값이 1/μ인 지수분포를 갖는다고 가정하였으며 (λ/μ) 값을 100에서 490까지 10 단위로 변화시키면서 시뮬레이션을 수행하였다.

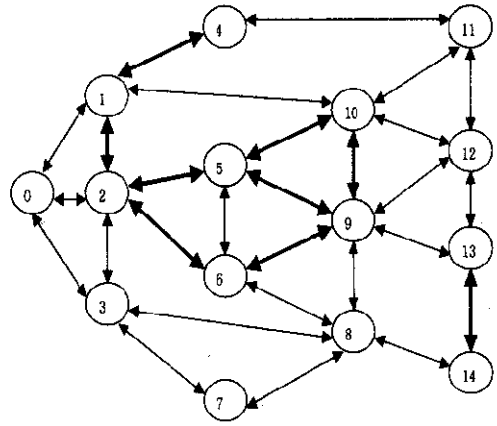


그림 2. 시뮬레이션에 사용된 네트워크 1
Fig. 2. Test Network 1

시뮬레이션 수행시 그림 2의 네트워크에서는 일반 링크의 용량(capacity)는 12, 굵게 표시된 링크의 용량은 48로 가정하였으며 그림 3과 4의 네트워크에서는 모든 링크의 용량은 48로 가정하였다. 이것은 실제 링크의 용량를 나타내는 것은 아니고 LSP setup 요구가 많이 차단되는 영역에서의 성능을 시뮬레이션 하기 위해서 링크의 residual capacity가 적게 남은 상태에서 성능 측정을 시작한 것을 의미한다.

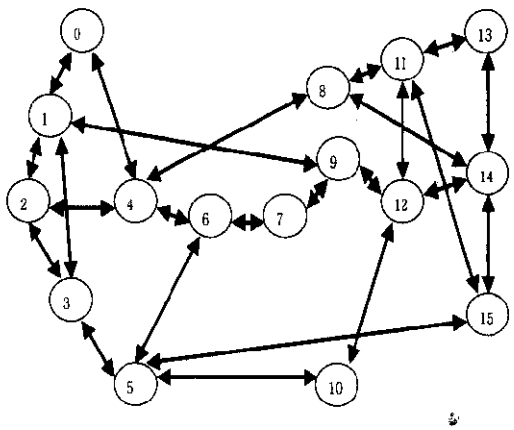


그림 3. 시뮬레이션에 사용된 네트워크 2
Fig. 3. Test Network 2

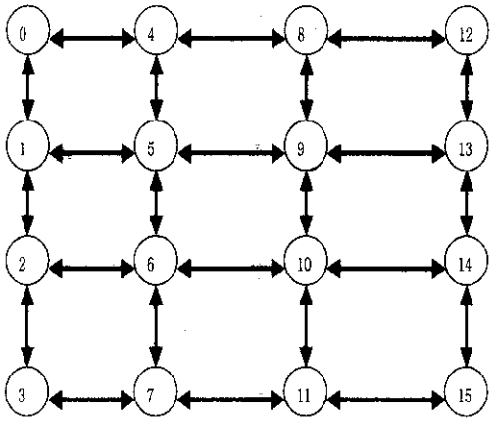


그림 4. 시뮬레이션에 사용된 네트워크 3
Fig. 4. Test Network 3

시뮬레이션을 위해서 모두 16개의 알고리즘들을 구현하였다. 각 알고리즘에서, shortest 는 minimum hop 을 widest 는 경로 대역폭을 기준으로 하였다. 따라서, 최단 경로 알고리즘(SP 알고리즘)은 최소 홉 경로를 구하게 된다. SP, SWP 그리고 WSP 알고리즘의 경우에는 별도의 대역폭 보장 방식이 존재하지 않기 때문에 알고리즘 실행 전에 요구되는 LSP 대역폭 보다 작은 link residual capacity를 갖는 링크를 네트워크에서 제외시키고 알고리즘을 실행시키는 방법을 사용하였다. 따라서 본 논문에서 고려하는 모든 알고리즘은 다 adaptive routing 알고리즘들이라고 할 수 있다.

제안된 알고리즘들에서 K 값은 1과 2인 두 경우를 고려하였으며 알고리즘 이름 뒤에 K1, K2를 붙여서 각각의 경우를 구분한다. 예를 들어서, WKMHK1 은 K = 1인 WKMH 알고리즘을 나타낸다. K 값이 1인 경우에는 하나의 최적 경로를 사용한다는 것이 아니라 앞에서 설명한 것처럼 모든 최적 경로를 사용한다는 의미가 된다. 마찬가지로 K 값이 2인 경우에도 2개의 path들 중에서 하나를 선택한다는 것이 아니라 각 기준에 따라서 모든 최적의 경로들과 그 바로 다음 단계의 모든 경로들을 구해서 그 중에서 하나의 경로를 선택한다는 것을 의미한다. DORA 알고리즘의 경우에는 [5]에서 성능이 가장 좋은 경우인 BWP = 0.9를 사용하였다.

시뮬레이션을 통해서 각 알고리즘별로 accepted bandwidth, accepted request number, average path hop이라는 3가지 성능 평가 지수를 측정하였다. 먼저, accepted bandwidth는 라우팅에 성공한 모든 LSP들의 대역폭을 더한 값을 나타내고 accepted request number는 라우팅에 성공한 총 LSP 개수를 마지막으로 average path hop은 라우팅에 성공한 모든 LSP들의 평균 hop 수로 계산된다.

표 1은 그림 2의 네트워크 1에서 네트워크 부하 (λ/μ) 값이 100에서 490까지 변할 때 각 알고리즘별로 accepted bandwidth를 보여준다. 표 1에서 알고리즘의 명칭을 다음과 같이 축약해서 표시하였다. WK1 = WKMHK1, WK2 = WKMHK2, RK1 = RKMUHK1, RK2 = RKMUHK2, SK1 = SKWK1, SK2 = SKWK2, MK1 = MUKMHK1, MK2 = MUKMHK2, DR = DORA, WMR = WSUMMIRA, LMR = LEXMIRA, PSM = MIPASMIRA, BSM = MIBLASMIRA. 부하를 x축으로 accepted bandwidth를 y축으로 하는 그래프의 형태로 결과를 표시하는 것이 일반적인 방식이겠지만 본 논문에서는 총 16개의 알고리즘들을 고려하였고 부하 값이 큰 상황에서 알고리즘들의 성능 차이가 크지 않았기 때문에 그래프 상의 그림으로는 각 알고리즘의 성능을 비교하기가 쉽지 않기 때문에 표 1과 같이 결과를 제시하였다.

표 1의 결과에서 첫 번째 행은 부하를 나타내고 그 밑의 두 열들이 해당 부하에 대한 결과를 나타낸다. 부하가 100일 때의 결과를 보면 WKMHK1 , WSP , RKMUHK1 , SKWK2 , MUKMHK2 , MUKMHK1 , SP , WKMHK2 , SWP , DORA , WSUMMIRA , LEXMIRA , MIPASMIRA , SKWK1 , MIBLASMIRA , RKMUHK2 순으로 accepted bandwidth 값이 감소하는 것을 알 수 있다.

표 1을 통해서 앞에서 예측한 것처럼 모든 부하에 대해서 계산이 복잡한 MIRA 계열 즉, WSUMMIRA, LEXMIRA, MIBLASMIRA, MIPASMIRA 알고리즘들이 가장 좋은 성능을 내지는 않는다는 것을 알 수 있다. 그것은 앞에서 언급한 것처럼 모든 노드들 사이에서 LSP가 설정될 수 있는 상황에서는 특별히 interference가 작은 경로가 존재할 가능성이 적으며 따라서 모든 노드들이 ingress-egress 노드 쌍이 되는 경우에는 MIRA 계열 알고리즘들의 성능이 하나의 ingress-egress 노드 쌍만을 고려하는 알고리즘들 보다 향상될 개연성이 적기 때문이다.

기존의 방식들[3][4][5][6]의 동적인 환경에서의 시뮬레이션 결과들을 보면 모두 작은 범위의 네트워크 부하에 대해서만 시뮬레이션을 수행하였고 그

범위 안의 부하에서 제안된 알고리즘들의 성능이 우수하다는 결론을 제시하고 있다.

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1
Table 1. Accepted bandwidth between load 100 and 490 - test network 1

100		110		120		130		140	
WK1	37259	WK1	34890	WK1	32440	MK1	30339	RK1	28693
WSP	37189	MK1	34658	WSP	32392	RK1	30252	WSP	28613
RK1	37170	SP	34553	MK2	32354	WSP	30227	WK1	28537
SK2	37056	WSP	34549	MK1	32303	SK2	30220	SK2	28530
MK2	37023	SK2	34541	SK2	32288	MK2	30091	MK2	28357
MK1	37000	MK2	34393	SP	32210	WK2	29974	WK2	28329
SP	36968	RK1	34364	RK1	32191	WK1	29968	MK1	28309
WK2	36799	WK2	34249	SK1	31648	WMR	29923	SP	28279
SWP	36632	WMR	33871	WMR	31642	SP	29841	WMR	28270
DR	36461	DR	33799	SWP	31608	SK1	29801	SK1	28147
WMR	36304	SWP	33785	WK2	31536	DR	29760	DR	27958
LMR	36234	SK1	33718	DR	31521	LMR	29617	LMR	27911
PSM	36044	LMR	33515	LMR	31225	SWP	29528	SWP	27882
SK1	36042	PSM	33440	PSM	31204	BSM	29213	PSM	27407
BSM	35859	BSM	33261	BSM	31141	PSM	29120	BSM	27404
RK2	32533	RK2	30366	RK2	28754	RK2	26634	RK2	25361
	36886		34541		32116		30036		28486

그러나 본 논문에서는 부하의 범위를 보다 넓게 변화시키면서 시뮬레이션을 수행하였고 그 결과인 표 1을 보면 모든 경우에 최적인 단일의 알고리즘이 존재하지 않는다는 것을 알 수 있다.

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

150		160		170		180		190	
MK1	27221	MK1	25769	RK1	24857	SK2	23399	SP	22567
SK2	27190	WSP	25755	SK2	24680	MK2	23374	WSP	22538
WK1	27000	SK2	25657	SP	24498	WK1	23335	MK1	22497
WSP	26944	RK1	25600	WK2	24447	WSP	23312	SK2	22485
SP	26913	MK2	25547	WSP	24419	SP	23295	MK2	22439
MK2	26912	WK2	25479	MK1	24368	RK1	23220	WK1	22386
WK2	26904	WK1	25463	MK2	24322	MK1	23158	SK1	22279
RK1	26901	SP	25422	WK1	24291	SK1	23132	RK1	22269
WMR	26731	SWP	25420	WMR	24242	WMR	23089	WK2	22145
SWP	26579	WMR	25298	DR	24085	WK2	23050	DR	22037
SK1	26505	SK1	25060	SK1	24038	SWP	23022	SWP	22005
DR	26411	BSM	25053	SWP	23930	LMR	22835	LMR	21974
LMR	26208	DR	25052	PSM	23857	DR	22758	WMR	21930
PSM	26091	LMR	24844	LMR	23778	BSM	22709	BSM	21885
BSM	25934	PSM	24669	BSM	23732	PSM	22498	PSM	21798
RK2	24147	RK2	23109	RK2	22015	RK2	20905	RK2	19997
	26949		25511		24608		23165		22341

이것은 어떻게 보면 자연스러운 결과인데 그 이유는 모든 알고리즘들이 heuristic하기 때문에 비록 동일한 네트워크 환경이라도 모든 노드들이 ingress-egress 노드 쌍이 될 수 있는 경우에는 부하에 따라서 약간씩 그 결과가 달라질 수 있기 때문이다.

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

200		210		220		230		240	
WK1	21720	WK1	20929	SK2	20233	SK2	19413	MK2	18884
MK1	21710	WSP	20859	WK1	20076	WSP	19391	SK2	18778
MK2	21655	SP	20809	SP	20026	MK1	19371	WSP	18712
RK1	21625	MK1	20711	MK2	20007	SP	19278	WK2	18653
SP	21491	MK2	20703	RK1	20000	MK2	19223	MK1	18610
SK2	21468	RK1	20672	WK2	19988	WMR	19195	RK1	18579
WSP	21464	SK2	20579	MK1	19955	WK1	19167	SP	18575
SWP	21315	DR	20513	WSP	19779	WK2	19164	SWP	18543
LMR	21138	SK1	20512	SK1	19734	SK1	19134	WK1	18432
DR	21137	WMR	20500	DR	19711	RK1	19112	WMR	18390
SK1	21127	WK2	20426	LMR	19665	SWP	19107	SK1	18371
WMR	21059	LMR	20404	SWP	19586	DR	18901	DR	18287
WK2	21022	SWP	20376	WMR	19584	LMR	18832	LMR	18200
PSM	20916	PSM	20189	BSM	19481	BSM	18783	PSM	18077
BSM	20798	BSM	20135	PSM	19447	PSM	18618	BSM	18011
RK2	19461	RK2	18747	RK2	18062	RK2	17547	RK2	16928
	21503		20720		20031		19219		18695

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

250		260		270		280		290	
WMR	18215	WK1	17800	MK1	17280	MK2	16711	WSP	16130
WSP	18183	WSP	17736	MK2	16999	WK1	16638	WK1	16101
SK2	18128	MK1	17622	RK1	16985	SK2	16580	SP	16066
MK1	18074	MK2	17478	WSP	16977	SP	16575	MK1	15972
SP	18041	SK2	17438	SP	16963	WSP	16469	SK1	15868
MK2	17973	SK1	17385	WK2	16956	MK1	16420	RK1	15864
WK1	17963	RK1	17313	SK2	16941	RK1	16404	MK2	15822
WK2	17954	WK2	17300	WK1	16784	SK1	16397	WK2	15793
LMR	17952	WMR	17256	WMR	16720	SWP	16369	SK2	15780
RK1	17910	SWP	17245	DR	16626	DR	16320	DR	15697
SK1	17839	PSM	17229	SK1	16594	WMR	16315	LMR	15670
SWP	17801	SP	17202	SWP	16587	LMR	16208	PSM	15665
DR	17705	DR	17092	LMR	16505	WK2	16189	SWP	15649
BSM	17705	LMR	17051	BSM	16450	BSM	16109	WMR	15617
PSM	17589	BSM	16896	PSM	16172	PSM	16027	BSM	15582
RK2	16277	RK2	16014	RK2	15541	RK2	15145	RK2	14811
	18033		17622		17107		16544		15969

모든 부하에 대해서 가장 성능이 좋은 하나의 알고리즘이 존재하지 않기 때문에 16개의 알고리즘들의 성능을 비교하거나 실제로 사용할 알고리즘을 선택

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

300		310		320		330		340	
WK1	15676	WSP	15267	SP	15113	WK1	14701	SP	14422
MK1	15647	WMR	15224	WSP	14962	WSP	14684	MK1	14359
WSP	15646	MK2	15206	WK1	14930	WK2	14639	WK1	14253
WK2	15609	RK1	15194	MK1	14888	RK1	14624	RK1	14197
SK2	15594	WK2	15190	MK2	14816	SP	14548	WSP	14150
RK1	15581	MK1	15189	SK1	14803	WMR	14532	SK1	14128
SP	15521	SK1	15119	WMR	14786	SK2	14525	MK2	14119
WMR	15511	WK1	15117	SK2	14738	MK1	14449	SWP	14114
MK2	15489	SWP	15089	RK1	14734	DR	14399	DR	14037
SWP	15484	SP	15064	DR	14706	MK2	14396	WK2	14022
SK1	15483	SK2	15053	WK2	14664	BSM	14244	SK2	13985
DR	15442	LMR	15044	PSM	14598	LMR	14224	WMR	13894
LMR	15407	DR	15036	BSM	14579	SWP	14171	PSM	13779
BSM	15237	PSM	15009	SWP	14577	PSM	14104	LMR	13749
PSM	14978	BSM	14769	LMR	14484	SK1	14022	BSM	13700
RK2	14247	RK2	14042	RK2	13691	RK2	13475	RK2	13016
	15519		15114		14962		14554		14278

해야 할 때 사용할 수 있는 새로운 성능 평가 지수가 필요한데 본 논문에서는 다음과 같은 방식으로 계산되는 최적 결과 비율(best results ratio)를 사용하였다.

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

350		360		370		380		390	
WSP	14036	WSP	13691	MK2	13345	WK2	13050	WSP	12831
DR	13917	SP	13601	SK2	13319	SP	13015	SK2	12809
WK1	13885	WK1	13593	WSP	13297	MK1	13000	RK1	12790
MK1	13799	MK2	13462	SWP	13264	WSP	12960	SP	12767
RK1	13793	SK2	13454	RK1	13230	RK1	12933	MK1	12696
SWP	13779	RK1	13439	LMR	13215	WMR	12922	MK2	12677
SK2	13775	LMR	13416	WK1	13187	WK1	12901	WK2	12608
SK1	13770	MK1	13410	SP	13137	SK2	12860	WK1	12556
SP	13757	SWP	13398	WK2	13081	MK2	12844	SK1	12524
WK2	13786	DR	13396	MK1	13044	SWP	12692	DR	12521
MK2	13781	SK1	13392	BSM	13042	LMR	12682	BSM	12506
LMR	13708	WK2	13355	DR	13034	DR	12666	LMR	12501
BSM	13616	WMR	13334	SK1	12966	SK1	12654	WMR	12463
WMR	13611	BSM	13225	PSM	12949	BSM	12603	SWP	12417
PSM	13540	PSM	13115	WMR	12949	PSM	12578	PSM	12405
RK2	12711	RK2	12483	RK2	12365	RK2	11967	RK2	11790
	13896		13554		13212		12920		12703

먼저 각 부하별로 최적의 결과를 기준으로 그 값의 1% 이내의 결과를 갖는 알고리즘들을 찾아서 그 알고리즘들을 해당 부하에서 최적 결과(best result)를 갖는 알고리즘들이라고 표시한다.

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

400		410		420		430		440	
WK1	12519	SK2	12352	WK1	12078	WK1	11847	WK1	11629
SK2	12509	MK1	12271	WK2	12032	WSP	11762	WK2	11603
MK1	12490	WSP	12210	MK1	12012	SP	11741	RK1	11593
SK1	12438	SP	12189	SK2	11977	SK2	11737	MK1	11511
SP	12359	BSM	12181	RK1	11858	MK1	11724	SK2	11508
WMR	12359	SK1	12160	WSP	11846	SWP	11695	SP	11496
SWP	12357	PSM	12119	MK2	11825	RK1	11686	SWP	11485
MK2	12340	SWP	12116	SP	11809	WMR	11647	MK2	11453
WK2	12329	WK1	12102	WMR	11792	WK2	11630	PSM	11384
WSP	12324	MK2	12054	SWP	11790	DR	11619	WSP	11378
RK1	12289	RK2	12033	SK1	11772	BSM	11544	DR	11376
PSM	12275	LMR	12031	DR	11740	PSM	11540	LMR	11372
DR	12265	WK2	12018	BSM	11669	LMR	11525	WMR	11363
BSM	12232	WMR	11960	LMR	11652	MK2	11521	SK1	11247
LMR	12020	DR	11928	PSM	11593	SK1	11499	BSM	11231
RK2	11425	RK2	11534	RK2	10974	RK2	10956	RK2	10618
	12394		12228		11957		11729		11513

모든 부하에 대해서 최적 결과를 갖는 알고리즘들을 찾은 후에 각 알고리즘 별로 최적 결과를 갖는 알고리즘으로 선택된 총 회수를 구해서 그 값을 총 부하의 개수로 나누면 그 값이 해당 알고리즘의 최적 결과 비율이 된다.

표 1. 부하 100 - 490에 대한 Accepted bandwidth - test network 1 (계속)
Table 1. Accepted bandwidth between load 100 and 490 - test network 1 (continued)

450		460		470		480		490	
WK1	11381	SWP	11172	WK1	11086	MK1	10955	RK1	10702
MK1	11374	SP	11140	MK1	10955	WSP	10878	WK1	10683
SP	11371	MK1	11136	MK2	10920	DR	10827	MK1	10644
SK2	11357	WK1	11118	WMR	10913	WMR	10810	WSP	10640
MK2	11347	MK2	11114	SK1	10908	WK1	10787	SP	10614
SWP	11257	WSP	11086	SP	10899	WK2	10756	WMR	10613
WK2	11250	DR	11059	SK2	10890	SP	10732	SK2	10606
WMR	11215	WK2	11033	RK1	10879	BSM	10701	MK2	10604
WSP	11209	SK1	11019	WK2	10842	SK2	10659	SWP	10602
RK1	11208	WMR	11016	SWP	10813	MK2	10659	WK2	10574
BSM	11199	SK2	10994	PSM	10791	SWP	10629	LMR	10507
SK1	11197	PSM	10930	DR	10784	SK1	10600	SK1	10503
LMR	11196	BSM	10927	WSP	10782	RK1	10560	DR	10485
DR	11165	RK1	10901	LMR	10667	LMR	10525	PSM	10474
PSM	11130	LMR	10774	BSM	10618	PSM	10486	BSM	10423
RK2	10595	RK2	10274	RK2	10251	RK2	10000	RK2	9953
	11267		11060		10975		10845		10595

이와 같은 최적 결과 비율은 알고리즘의 선택 기준으로 사용할 수 있다. 즉, 최적 결과 비율이 클수록 많은 부하에서 좋은 성능을 갖는 알고리즘을 나타

낸다고 할 수 있다.

표 2. 최적 결과 비율 - test network 1
Table 2. Best results ratio - test network 1

Algorithm	Accepted BW	Algorithm	Avg. path hop	Algorithm	Accepted req. num.
WSP	0.66	WSP	0.78	WK1	0.78
WK1	0.63	WK1	0.76	WSP	0.73
SK2	0.59	MK1	0.73	MK1	0.66
MK1	0.59	RK1	0.63	DR	0.63
SP	0.51	SP	0.63	SK2	0.59
RK1	0.39	SK2	0.63	MK2	0.56
MK2	0.39	MK2	0.49	RK1	0.51
WK2	0.17	WK2	0.24	SP	0.51
WMR	0.12	SK1	0.17	WK2	0.44
SWP	0.07	SWP	0.07	SWP	0.32
SK1	0.05	WMR	0.07	WMR	0.32
DR	0.02	DR	0.02	SK1	0.27
LMR	0.02	LMR	0.00	BSM	0.07
BSM	0.00	BSM	0.00	LMR	0.07
PSM	0.00	PSM	0.00	PSM	0.05
RK2	0.00	RK2	0.00	RK2	0.00

여기서 최적의 값 하나가 아니라 최적의 1% 이내의 값들을 기준으로 한 것은 시뮬레이션 오차를 고려한 것으로 1% 이외의 기준이 적용될 수 있을 것이다. 여기서는 한 예로 1%를 적용한 것이다.

표 3. 최적 결과 비율 - test network 2
Table 3. Best results ratio - test network 2

Algorithm	Accepted BW	Algorithm	Avg. path hop	Algorithm	Accepted req. num.
MK2	0.80	MK2	0.85	MK2	0.88
WK1	0.68	WK1	0.76	RK1	0.83
MK1	0.66	WSP	0.76	MK1	0.76
RK1	0.63	RK1	0.76	WK1	0.76
SP	0.61	SP	0.76	WSP	0.73
SK2	0.59	MK1	0.73	SK2	0.71
WSP	0.59	SK2	0.66	WK2	0.56
WK2	0.39	WK2	0.44	SP	0.54
WMR	0.02	SK1	0.02	DR	0.41
SK1	0.00	SWP	0.02	WMR	0.15
RK2	0.00	DR	0.02	SK1	0.10
SWP	0.00	WMR	0.02	SWP	0.07
DR	0.00	RK2	0.00	LMR	0.02
LMR	0.00	LMR	0.00	RK2	0.00
PSM	0.00	PSM	0.00	PSM	0.00
BSM	0.00	BSM	0.00	BSM	0.00

예를 들어서 표 1에서 부하가 100인 경우를 살펴보자. 최적 값은 WKMHK1 알고리즘의 37259이고 그 값의 1% 이내에 있는 알고리즘들은 37259x(1-0.01) = 36886.41 보다 큰 값을 갖는 알고리즘들 즉, WKMHK1, WSP, RKMUHK1, SKWK2, MUKMHK2, MUKMHK1, SP 알고리즘들이 부하 100에서 최적 결과를 갖는 알고리즘들이 된다. 이와 같은 과정을 부하 110, ..., 490까지 반복해서 구하면 모든 부하들에 대한 최적 결과 알고리즘들을 구할 수 있다. 그 후에 모든 알고리즘들에 대해서 각 알고리즘별로 다시 부하 100부터 490까지 최적 결과 알고리즘에 속하는 총 회수를 구해서 전체 부하 개수 40로 나누면 표 2와 같은 결과를 얻을 수 있다.

표 4. 최적 결과 비율 - test network 3
Table 4. Best results ratio - test network 3

Algorithm	Accepted BW	Algorithm	Avg. path hop	Algorithm	Accepted req. num.
WSP	0.76	WSP	0.85	RK1	0.83
RK1	0.68	RK1	0.80	WSP	0.76
WK1	0.66	MK2	0.78	MK2	0.73
WK2	0.66	MK1	0.76	MK1	0.73
MK2	0.61	WK1	0.76	SK2	0.68
MK1	0.61	WK2	0.76	WK1	0.61
SK2	0.54	SK2	0.73	WK2	0.61
SP	0.44	SP	0.54	DR	0.56
WMR	0.05	SWP	0.05	SP	0.24
SK1	0.05	WMR	0.05	SK1	0.24
DR	0.02	DR	0.02	WMR	0.12
RK2	0.00	SK1	0.00	SWP	0.07
SWP	0.00	RK2	0.00	PSM	0.05
LMR	0.00	LMR	0.00	LMR	0.02
PSM	0.00	PSM	0.00	BSM	0.02
BSM	0.00	BSM	0.00	RK2	0.00

표 2의 결과는 그림 2의 네트워크 1에 대한 시뮬레이션 결과에서 표 1에 보인 accepted bandwidth 뿐 아니라 accepted request number와 average path hop에 대한 최적 결과 비율을 보이고 있다. 여기서, average path hop의 경우에는 다른 두 값들과는 달리 가장 작은 값과 그 값에 1% 이내에 드는 값을 기준으로 계산된 것이다.

표 2의 accepted bandwidth에 대한 결과를 보면 WKMH, WSP, MUKMH 그리고 SP 알고리즘과 같이 최소 홉 경로에 기초한 알고리즘들의 성능이 좋은 것을 볼 수 있다. 이것은 모든 노드들이 ingress-egress 노드 쌍이 될 수 있는 환경에 대해서 고려하고 부하가 큰 경우에는 가급적 홉 수가 적은 경로를 고려하는 것이 미래의 LSP setup 요구에 대한 interference를 작게 하는 효율적인 방법이 된다는 것을 의미한다고 할 수 있다.

그것은 표 2에서 average path hop에 대한 결과를 통해서 확인할 수 있다. 표 2의 결과를 보면 average path hop에 대한 결과가 accepted

bandwidth의 결과와 거의 일치하는 것을 볼 수 있다. 즉, average path hop이 적은 경로를 선택하는 알고리즘들이 accepted bandwidth 측면에서 유리하다는 것이다.

예외적으로 SKWK2 알고리즘의 성능이 좋게 나오는데 그것은 SKWK2 알고리즘의 성능은 좋지만 SKWK1의 성능은 그 보다 떨어지는 것으로 보아서 widest 기준 보다는 shortest 기준에 더 영향을 받은 것으로 생각된다. 즉, SKW 알고리즘에서는 K = 2 일 때가 K = 1 일 때보다 더 많은 후보 경로들 중에서 최소 홉을 갖는 경로를 택하기 때문에 K = 2 일 때가 K = 1 일 때 보다 항상 더 작거나 혹은 최소한 동일한 홉을 갖는 경로를 택할 수 있고 그 결과 accepted bandwidth 성능이 더 좋아지는 것이다. 이것은 표 2의 average path hop 결과로도 확인할 수 있다.

비슷하게 RKMUH 알고리즘의 결과도 설명될 수 있다. RKMUH 알고리즘은 K = 1 일 때가 K = 2 일 때보다 성능이 좋은데 그것은 RKMUH 알고리즘은 후보 경로들 중에서 랜덤하게 하나를 선택하기 때문에 후보 경로가 더 많은 K = 2 일 때가 후보 경로가 적은 K = 1 일 때보다 더 큰 홉을 갖는 경로를 선택할 가능성이 크기 때문이다. 물론, K = 2 인 경우가 후보 경로들의 개수가 더 많고 따라서 홉 수가 작은 경로들이 더 많이 포함될 가능성도 있다. 그러나, RKMUH 알고리즘은 기본적으로 링크 가중치를 링크 사용율(link utilization)로 해서 최단 거리를 갖는 경로를 구하는 방식 즉 경로를 구성하는 링크들의 링크 가중치 합이 최소가 되는 경로를 구하는 것이므로 링크의 가중치가 작을 뿐만 아니라 경로의 홉 수가 작을수록 RMMUH 알고리즘의 최적 경로로 선택된다. 따라서, K = 2 일 때가 K = 1 일 때보다 홉 수가 더 작은 경로들을 포함할 가능성 보다는 오히려 홉 수가 더 큰 경로들을 포함할 가능성이 더 크다고 할 수 있으며 이런 이유로 RKMUHK1의 성능이 RKMUHK2 보다 더 좋아진다고 할 수 있다. 이것 역시 표 2의 average path hop 결과로 확인할 수 있다.

표 2에서 보이는 accepted request number에 대한 결과는 앞에서 살펴본 accepted bandwidth의 경우와 유사하지만 약간의 차이점이 존재하는 것을 알 수 있다. 그것은 이 결과에는 각 LSP의 대역폭이 반영되지 않으므로 대역폭 1인 LSP setup을 10개 받아들인 알고리즘이 대역폭 3인 LSP 4개를 받아들인 알고리즘보다 accepted request number 측면에서는 더 좋아지기 때문이다.

특기할 만한 것은 DORA 알고리즘으로 대역폭 1인 LSP setup 요구에 대한 성능이 아주 좋기 때문에 accepted bandwidth 성능에 비해서 accepted request number 성능이 매우 높게 나오는 것이라고 설명할 수 있다. WSUMMIRA 알고리즘의 경우에도 대역폭이 작은 LSP setup 요구에 대해서는 어느 정도 효과가 있음을 알 수 있다. 그러나, 그 계산의 복잡도를 고려해 보면 이와 같은 성능 향상이 큰 의미가 있다고는 판단되지 않는다.

표 3은 그림 3의 네트워크 2에 대한 시뮬레이션 결과에서 계산된 accepted bandwidth, accepted request number와 average path hop에 대한 최적 결과 비율을 보이고 있다. 표 2의 결과와 마찬가지로 MUKMH, WKMH 알고리즘과 같이 최소 홉 경로에 기초한 알고리즘들의 성능이 좋다는 것을 알 수 있다.

특기할 만한 것은 RKMUH 알고리즘으로 랜덤한 선택을 통해서 오히려 설정된 LSP들의 average path hop이 작아지게 되었고 그 결과 accepted bandwidth나 accepted request number 성능이 좋아지게 되었다고 설명할 수 있다. 그러나, 이것은 표 2의 결과와 마찬가지로 K = 1인 경우에 해당되는 것이고 K = 2인 경우에는 오히려 홉이 큰 경로들을 선택할 가능성이 커져서 성능이 나빠지게 된 것으로 생각된다.

표 4는 그림 4의 네트워크 3에 대한 시뮬레이션 결과에서 계산된 accepted bandwidth, accepted request number와 average path hop에 대한 최적 결과 비율을 보이고 있다. 표 2나 3의 결과와 마찬가지로 WSP, MUKMH 그리고 WKMH 알고리즘과 같이 최소 홉 경로에 기초한 알고리즘들의 성능이 좋다는 것을 알 수 있다. 한편, 표 3에서처럼 RKMUHK1 알고리즘의 성능이 좋은데 그것은 역시 average path hop의 성능이 좋기 때문인 것을 표 4에서 알 수 있다.

3개의 네트워크에 대한 이상의 시뮬레이션 결과를 종합해 보면 첫째로는 MIRA 알고리즘이 계산의 복잡성에 비해서 성능이 그리 뛰어나지 못하다는 점을 확인했고 둘째로는 넓은 범위의 부하를 고려하는 경우 모든 부하에서 가장 좋은 성능을 갖는 단일의 알고리즘이 존재하지 않는 다는 것도 알 수 있다.

셋째로는 제안된 알고리즘들이 세 경우 모두에서 일정 수준이상의 성능을 보임을 확인할 수 있다. 특

히, WKMHK1 알고리즘이 세 경우 모두에서 accepted bandwidth 측면이나 accepted request number 측면에서 큰 기복이 없이 항상 일정 수준 이상 - 예로 0.6 이상의 성능을 보임을 알 수 있다. 이것은 WSP 알고리즘의 경우도 비슷한데 결론적으로 경로를 구하는 다양한 기준들 중에서 shortest 중에서 widest 경로를 택하는 방식이 여러 상황에서도 일정 수준 이상의 성능을 낼 수 있는 알고리즘이라고 할 수 있다.

마지막으로 제안된 알고리즘에서 K 값의 영향을 생각해 보면 다음과 같다. 앞에서도 언급하였지만 본 논문에서 제안된 k shortest loopless path 알고리즘은 k개의 shortest loopless path를 구하는 것이 아니라 k번째 수준에 포함되는 모든 경로들을 구하기 때문에 알고리즘에서 K = 2 일 때 구하게 되는 경로들의 수는 K = 1 일 때 구하는 것에 비해서 무척 많아진다. 이런 이유로 K = 2 까지만 시뮬레이션을 수행한 것이다.

이제, K 값에 따른 성능 변화를 살펴보면 먼저, MUKMH 알고리즘의 경우에는 MUKMHK1 알고리즘이 세 경우 모두에서 일정 수준 이상의 성능을 보임을 알 수 있다. MUKMHK2 알고리즘의 경우에는 표 2의 accepted bandwidth 의 경우를 제외하고는 모두 성능이 좋게 나오지만 편차가 있는 것을 볼 수 있다. 그것은, 항상 K = 2 일 때가 K = 1 일 때보다 더 큰 홉 수를 갖는 후보 경로들을 포함하기 때문에 average path hop 성능이 더 나빠질 수 있는 가능성이 크기 때문이다. 물론, 반대로 더 좋아질 수도 있다. 그러나, 네트워크 토폴로지나 부하에 따라서 성능이 가변적인 알고리즘보다는 항상 일정 수준 이상의 성능을 낼 수 있는 것이 더 바람직하다고 생각되며 따라서, 세 경우 모두에서 일정한 성능을 갖는 MUKMHK1 알고리즘이 MUKMHK2 알고리즘 보다 더 실용적이라고 할 수 있을 것이다.

WKMH 알고리즘의 경우에는 K = 2 일 때의 성능이 K = 1 일 때의 성능보다 훨씬 떨어지므로 MUKMH 알고리즘처럼 K = 1 이면 충분하다고 할 수 있다. 이외에도 RKMUH 알고리즘의 경우에도 앞에서 언급한 것처럼 K = 2 일 때의 성능이 나쁘므로 역시 K = 1 이면 충분하다. 마지막으로 SKW 알고리즘의 경우에는 역시 앞에서 살펴본 것처럼 K = 2 일 때의 성능이 좋다. 그러나, SKWK2 알고리즘의 성능은 WKMHK1이나 MUKMHK1 알고리즘의 성능보다 떨어지므로 제안된 방식들 중에서는 계산량이 가장 적은 WKMHK1과 MUKMHK1 알고리즘이 가장 실용적인 알고리즘이라고 할 수 있다.

Ⅳ. 결론

본 논문에서는 MPLS 네트워크에서 LSP 설정에 적용될 수 있는 대역폭 보장 온라인 라우팅 알고리즘들을 제안하였다. 제안된 알고리즘들은 K-shortest loopless path 알고리즘에 기초해서 기존의 WSP와 SWP 알고리즘을 확장한 형태를 갖는다. 기존의 대역폭 보장 온라인 알고리즘들 중에서 성능이 좋다고 알려진 MIRA 계열의 알고리즘들은 전체 노드 쌍 중에서 일부만이 LSP의 ingress-egress 노드 쌍으로 사용되는 경우에는 성능이 좋지만 전체 노드 쌍을 모두 고려할 때는 계산의 복잡도에 비해서 성능이 그리 좋아지지는 않았다. 그것은 minimum interference 라는 목적은 최선이지만 실제로 그것을 찾아낼 수 있는 최적의 알고리즘이 제시된 것은 아니기 때문에 모든 노드 쌍을 고려하는 경우에는 제시된 heuristic 방식들이 그리 큰 효과를 내지 못한 것으로 판단된다. 모든 노드 쌍을 고려하고 부하가 큰 경우에는 최소 홉에 기초한 알고리즘들인 WKMH, MUKMH 그리고 WSP 알고리즘의 성능이 좋았다. 이것은 더 많은 시험 네트워크들을 통해서 검증을 해야 할 필요가 있지만 기본적으로는 모든 노드 쌍들이 최소 홉 경로를 사용하려고 하는 것이 heuristic하게 미래의 LSP 설정에 대한 interference를 줄이는 효과를 내기 때문이라고 할 수 있다.

각 링크의 대역폭 및 네트워크 토폴로지 정보는 기존의 라우팅 알고리즘을 크게 수정하지 않아도 비교적 정확한 값을 쉽게 얻을 수 있기 때문에 그와 같은 정보만으로 빠르게 대역폭 보장 라우팅을 수행할 수 있는 최소 홉 기반의 알고리즘들이 실용적인 측면에서 우수하다고 결론지을 수 있을 것이다.

특히, 제안된 방식들 중에서 WKMHK1과 MUKMHK1 알고리즘은 계산이 빠르면서도 일정 수준 이상의 성능을 낼 수 있는 방식으로 실용성이 높다고 생각된다. 추후에는 좀더 다양한 형태의 네트워크에서 성능을 평가하고 대역폭 이외의 제한 조건을 갖는 다중 제한 조건 라우팅 알고리즘에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] E. Rosen, et al., Multiprotocol Label Switching Architecture, RFC 3031, Jan. 2001

[2] G. Apostolopoulos, et al., QoS Routing Mechanisms and OSPF Extensions, RFC 2676, Aug. 1999

[3] Zheng Wang, et al., "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE JSAC*, Vol. 14, No. 7, pp. 1228-1234, Sep. 1996

[4] Koushik Kar, et al., "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE JSAC*, Vol. 18, No. 12, pp.2566-2579, Dec. 2000

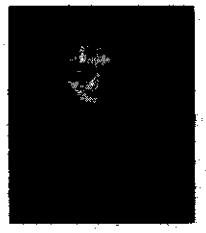
[5] W. Szeto, et al., "Dynamic Online Routing Algorithm for MPLS Traffic Engineering," *LNCS 2345*, pp. 936-946, 2002, Springer-Verlag

[6] Ilias Iliadis, et al., "A New Class of Online Minimum-Interference Routing Algorithms," *LNCS 2345*, pp. 959-971, 2002, Springer-Verlag

[7] E. Q. V. Martins, et al., "A NEW IMPLEMENTATION OF YEN'S RANKING LOOPLESS PATHS ALGORITHM," Technical Report 01/003, CISUC, Oct. 2000

[8] R. K. Ahuja, et al., *Network Flows: Theory, Algorithms, and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1993

이 준 호(Jun-Ho Lee) 정회원



1988년 2월: 연세대학교 전자공학과 (공학사)
1990년 8월: 연세대학교 전자공학과 (공학석사)
1996년 3월: 연세대학교 전자공학과 (공학박사)
1998년 4월 ~ 현재: 서울산업대학교 전자정보공학과 조교수

〈주관심분야〉 컴퓨터통신, 광통신

이 성 호(Seong-Ho Lee) 정회원

1980년 2월: 한국항공대학 전자공학과 (공학사)
1989년 2월: 한국과학기술원 전기 및 전자공학과 (공학석사)
1993년 2월: 한국과학기술원 전기 및 전자공학과 (공학박사)
1995년 3월 ~ 현재: 서울산업대학교 전자정보공학과 부교수

〈주관심분야〉 광통신, 컴퓨터통신