# 메쉬 구조의 망에서 대역 공유를 통한 복구 경로 설정 알고리즘

이황규*.홍석원*

## Backup path Setup Algorithm based on the Bandwidth Sharing in Mesh Networks

Hwang-Kyu Lee*.Sug-Won Hong*

이 논문은 2001년도 명지대학교 학술 연구비의 지원을 받아 연구되었음

요 약

망 복구에 있어서 주요한 과제는 작업 경로에 대한 복구 경로를 설정하는 것이다. 장애가 발생하더라도 서비스의 중단이 없도록 하기 위해서는 복구 경로를 설정할 때 작업 경로의 대역을 보장할 수 있는 복구 경로를 설정해야 한다. 따라서 이 과제는 대역을 보장하는 작업 경로와 복구 경로를 동시에 설정하는 QoS 라우팅의 해결책이 요구된다. 본 논문에서는 특정 복구 경로가 복구의 대상으로 하는 작업 경로와 동일 링크를 지나지 않는 작업 경로에 대한 다른 복구 경로들과 대역을 공유함으로써 최대한 대역의 소비를 피하면서, 작업 경로의 서비스를 보장할 수 있는 복구 경로를 설정하는 알고리즘을 제시하고 시뮬레이션을 통해서 복구 경로로 인한 대역의 소비 효과와 거부 효과를 분석하였다.

ABSTRACT

Path setup considering QoS is one of the main problems to solve in Internet traffic engineering. In network restoration that is one of the traffic engineering components, to establish a backup path or backup path for a working path to be protected is the main task. The backup path should have enough bandwidth to guarantee the working paths. Thus this work requires the QoS routing solution that is to set up a backup path as well as a working path at the same time provisioning enough bandwidth. In this paper, we propose an algorithm to setup a backup path that shares the bandwidth of other backup paths whose working path does not pass along the same link with the working path we are considering. In this way we can reduce much bandwidth consumption caused by setting up backup paths. We also analyze the disjoint path computation algorithms. Finally we show simulation results how the algorithm can reduce the bandwidth consumption and how it will affect blocking when we setup paths.

* 명지대학교 정보통신공학과

접수일자: 2002. 2. 19

## Ⅰ. 서 론

망의 복구는 망의 신뢰성을 결정하는 중요한 요인으로 공중망과 같은 주요한 망에서는 주요한 기능으로 제공되어 왔다. 최근 인터넷의 경우에도 점차 동기식 혹은 WDM 기반의 고속 트렁크 선로의 설치로 인해 링크의 장애가 발생할 경우 수많은 가입자 트래픽에 영향을 주게 되어 망 복구 기능에 대한 고려가 높아지고 있다. 현재까지의 망 복구 기능은 주로 SDH/SONET 링 구조의 보호 기능에 의존하고 있다. 하지만 최근 서비스 차원의 복구를 위해서 상위 계층에서 일반적인 메쉬 구조의 망에서 논리 채널 단위의 복구 보호에 대한 연구가 진행되고 있다[1,2].

일반적인 메쉬 토폴로지에서의 망 복구 방안으로는 여러 가지 방안이 제시되어 왔다[3]. 하지만 대부분의 망 복구 알고리즘은 장애가 발생하더라도 서비스를 보장할 수 있는 대역에 대한 고려는 하고 있지 않다. 복구 경로(backup path)를 설정할 때 원래의 작업 경로(working path)의 서비스 요구 조건을 충족시키기 위해서는 작업 경로에서 보장한 대역을 역시 보장할 수 있는 복구 경로를 설정해야 한다. 따라서 이 과제는 대역의 보장을 고려하여 경로를 설정하는 QoS 라우팅의 해결책이 요구된다.

대역을 보장하는 복구 경로를 설정할 때 모든 작업 경로에 대한 복구 경로의 대역을 별도로 할당한다면 복구 경로의 설정으로 인한 대역의 낭비가 심해지게 된다. 따라서 복구 경로의 설정에 있어서 작업 경로의 대역을 공유하면서 경로를 설정하게 되면 위의 경우보다 훨씬 대역의 소비를 감소시킬 수 있다[4,5]. [4]에서는 복구 경로의 공유 알고리즘을 무 공유(no sharing), 완전 공유, 단순 공유의 세 가지 경우로 구분하여 비교한 후 단순 공유의 알고리즘을 제안하고 있다. 하지만 단순 공유의 알고리즘의 경우 공유를 하지 않는 경우와 비교할 때 망의 토폴로지에 따라서 소비 대역이 증가하는 모순을 발견할 수 있다[6]. 따라서 본 논문에서는 작업 경로를 설정하면서 동시에 망 복구를 위한 복구 경로를 설정할 때 단순 공유의 약점을 보완하면서 복구 경로로 인한 대역의 낭비를 최소한 감소할 수 있는 알고리즘을 제안한다.

논문의 구성은 다음과 같다. 다음 절에서는 대역을 보장하는 복구 경로 설정의 문제를 정의한다. 그리고 3절에서는 이러한 문제를 해결하면서 대역을 보장하는 망 복구 경로를 설정하는 알고리즘에 대해서 설명한다. 4절에서는 이 알고리즘에서 사용하고 있는 디스조인트 경로 계산(disjoint path computation) 알고리즘에 대해서 시뮬레이션 결과를 통해 분석하고, 그리고 대역 소비와 거부에 대한 결과를 시뮬레이션을 통해서 구하여 알고리즘의 대역 감소 효과를 살펴본다.

## Ⅱ. 대역을 보장하는 복구 경로 설정 문제의 정의

대역을 포함한 일반적인 조건(constraint)을 보장하는 경로 설정의 문제는 다음과 같이 나타낼 수 있다. 하나의 시작 노드에서 하나의 종점 노드에 이르는 트래픽 흐름이 주어져 있을 때 이 트래픽을 물리적 망에서 요구 조건을 충족하는 경로를 찾아서 이를 할당하는 것이다. 그림 1에서 보이는 바와 같이 시작 노드 1에서 종점 노드 6에 이르는 트래픽 트렁크(traffic trunk)가 존재한다. 트래픽 트렁크는 하나의 서비스 망 도메인 내에서 동일한 경로를 갖고 또한 동일한 서비스 요구 사항을 갖는 통합된 트래픽 흐름(aggregate traffic flow)을 의미한다. 이러한 트래픽 트렁크의 속성(attribute)으로는 트래픽 파라메터, 우선 순위, 선점(preemption), 자원 클라스의 존재 여부(resource class affinity) 등의 속성을 갖을 수 있다[7].

이러한 트래픽 트렁크를 위해 경로를 배정하게 되는 물리적인 망은 노드 1과 6를 연결하는 여러 교환 노드와 노드 사이의 링크로서 구성되어 있다. 물리적인 망은 자원 속성(resource attribute)을 갖는데, 주로 링크의 자원 속성으로 구성된다. 예를 들면 각 링크에서 우선 순위 트래픽 트렁크에 현재 할당할 수 있는 가용 대역의 크기, 각 링크에 대해서 서비스 망 사업자가 부여한 정책 속성 등을 들 수 있다.

망의 보호를 위해서 링크의 장애가 발생하더라도 중단 없이 서비스가 제공되기 위해서는 작업 경로에 대한 복구 경로를 설정해야 한다. 이러한 조건을 충족하는 경로 설정의 과제는 작업 경로의 설정 뿐 아니라 복구 경로의 설정에도 그대로 적용된다. 즉 작업 경로의 요구 조건을 그대로 충족하는 복구 경로를 설정하여야 망 장애의 발생 시 서비스의 중단을 피할 수 있

다. 또한 복구 경로는 작업 경로와는 링크를 공유하지 않아야 한다는 조건을 충족해야 한다. 그림1의 예에서 만약 작업 경로 1-3-6, 1-4-3-6과 1-4-5-6의 세 개의 작업 경로가 설정될 때 이 경로들의 복구 경로는 작업 경로와 링크를 공유하지 않는 디스조인트 경로로서 1-2-6을 설정하게 된다. 이 복구 경로는 원래의 작업 경로의 대역을 보장할 수 있는 대역을 갖고 있어야 한다. 본 논문에서는 복구 경로를 설정할 때 다음의 조건을 충족하는 알고리즘을 제안한다.

첫째로 작업 경로의 대역을 보장할 수 있는 복구 경로를 설정한다. 이때 동일한 링크를 공유하는 작업 경로를 보호하는 복구 경로들 간에는 서로 대역을 공유하지 않는다. 하지만 동일한 링크를 공유하지 않는 모든 작업 경로에 대한 복구 경로들은 서로 대역을 공유하도록 하여 대역의 낭비를 최소한으로 하도록 하였다.

두 번째로 작업 경로와 복구 경로는 동일한 시작 노드에서 시작하여 동일한 종단 노드에서 경로가 끝나며 경로 상에서는 어떠한 링크도 서로 공유하지 않도록 하였다. 이것은 링크의 장애가 발생하였을 때 안전한 복구 경로를 보장하기 위한 필요 조건이라고 할 수 있다.

그림 1의 예에서 작업 경로 1-3-6, 1-4-3-6과 1-4-5-6의 세 가지 경로에 대해서 대역을 보장하는 복구 경로로서 1-2-6의 경로를 선택할 때 1-3-6과 1-4-5-6은 서로 링크를 공유하지 않으므로 복구 경로는 대역을 공유할 수 있다. 하지만 1-3-6과 1-4-3-6의 경로는 링크 3-6을 서로 공유하고 있으므로 1-2-6의 복구 경로에서 서로 대역을 공유하지 않는다. 따라서 만약 링크 3-6에서 장애가 발생할 경우 두 작업 경로는 1-2-6에 이르는 경로를 통해 서비스를 보장 받을 수 있다.

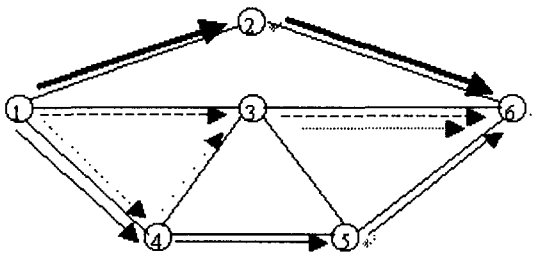


그림1: 작업 경로와 복구 경로의 예
Fig 1: Example of working paths and backup paths

## III. 대역 보장을 위한 복구 경로 설정 알고리즘

본 논문에서는 작업 경로의 대역 요구 조건을 만족하면서 동시에 망 장애가 발생시 작업 경로의 서비스를 지장 없이 제공할 수 있도록 하기 위해서 작업 경로의 대역을 고려한 복구 경로를 설정하는 알고리즘을 제시한다. 이러한 조건을 만족하면서 작업 경로와 복구 경로를 설정하는 알고리즘은 다음과 같다.

먼저, 주어진 망 토폴로지에서 대역 요구 조건을 만족하는 최단 거리의 작업 경로를 구한다. 이 때 각 링크에서 필요한 대역은 작업 경로의 요구 대역이 된다. 복구 경로는 대역의 낭비를 줄이기 위해서 가능한 범위에서 복구 경로를 위해 할당하는 대역은 공유하도록 한다. 이러한 경우, 각 링크에서 복구 경로에 대한 요구 대역은 현재 링크에서 복구 경로들 간에 공유된 대역의 크기와 복구 경로에 대한 작업 경로가 어떠한 링크로 구성되어 있는가에 따라서 달라진다.

만약 요구 대역이 대체 링크 상에 이미 할당되어 있는 복구 경로의 대역보다 작다면 다른 복구 경로와 대역을 공유하기 때문에 더 이상 대역을 할당할 필요가 없다. 요구 대역이 복구 경로 상에 이미 설정되어 있는 복구 경로의 대역보다 크다면, 요구 대역에서 이미 설정되어 있는 복구 경로의 대역을 뺀 만큼의 대역이 필요하다.

어떤 작업 경로에 대한 복구 경로의 대역 설정 시 동일한 링크를 지나가고 있는 다른 작업 경로에 대한 복구 경로가 같은 복구 경로 상의 링크에 존재한다면, 두 복구 경로의 대역은 서로 공유하지 않는다. 만약 이러한 두 개 이상의 복구 경로가 서로 대역을 공유할 경우 두 복구 경로의 작업 경로가 동시에 지나가는 링크에 장애가 발생한다면, 원래의 작업 경로에 대한 대역을 보장할 수 없기 때문이다. 다음은 이러한 알고리즘을 기술하고 있다.

$C_{ij}$ : 링크 (i,j)의 최대 대역

$m^k$ : 작업 경로 k의 요구 대역

$a_{ij}^k$ : 링크(i,j)를 지나가는 작업 경로 k의 필요 대역

$b_{ij}^k$ : 작업 경로 k에 대해서 링크(i,j)를 지나가는

복구 경로의 필요 대역

$$A_{ij}=\sum_k a_{ij}^k \quad B_{ij}=\sum_k b_{ij}^k$$

$n_{uv}$ : 링크(u,v)에서 복구 경로의 요구 대역

$l_{ij}^{uv}$ : 링크(u,v)를 지나가는 복구 경로의 대역의 합, 이러한 복구 경로의 작업 경로는 링크(i,j)를 통과한다.

$n_{ij}^k$ : 링크(i,j)를 지나가는 작업 경로 k에 대한 복구 경로의 요구 대역

초기값: $A_{ij}=0,\ B_{ij}=0,\ R_{ij}=C_{ij}$

1단계: 다음의 조건을 만족하는 최단 경로의 작업 경로를 계산한다.

$$a_{ij}^k \le R_{ij} \quad \text{for all link(i,j)}$$

여기서

$$a_{ij}^k=\begin{pmatrix} m^k & \text{if } R_{ij} > m^k \\ \infty & \text{if } R_{ij} < m^k \end{pmatrix}$$

2단계: 링크 속성(link attribute), 즉 가용 대역을 다시 계산한다.

$$R_{ij}=\begin{pmatrix} R_{ij}-a_{ij}^k & \text{if } R_{ij} > a_{ij}^k \\ R_{ij} & \text{if } R_{ij} < a_{ij}^k \end{pmatrix}$$

$$A_{ij}=A_{ij}+a_{ij}^k \text{ for all (i,j)}$$

3단계: 작업 경로 k에 대한 복구 경로가 요구하는 대역을 각 링크(u,v)에서 계산한다. 여기서 p를 1단계에서 계산한 작업 경로가 지나가는 링크의 집합이라고 하자.

$$n_{uv}^k=\begin{pmatrix} l_{ij}^{uv}=l_{ij}^{uv}+m^k & \text{if } l_{ij}^{uv}\neq 0,(i,j)\in P \\ m^k & otherwise \end{pmatrix}$$

4단계: 다음의 조건을 만족하는 복구 경로를 계산한다. 이 경로는 작업 경로와 겹치지 않는 최단 경로가 된다.

$$b_{ij}^k < R_{ij} \quad \text{for all (i,j)}$$

여기서,

$$b_{ij}^k=\begin{pmatrix} 0 & \text{if } n_{ij}^k < B_{ij} \\ n_{ij}^k-B_{ij} & \text{if } n_{ij}^k > B_{ij} \text{ and } R_{ij}\ge n_{ij}^k-B_{ij} \\ \infty & \text{if } R_{ij} < n_{ij}^k-B_{ij} \end{pmatrix}$$

5단계: 링크 속성, 즉 링크 가용 대역을 새로 계산한다.

$$R_{ij}=\begin{pmatrix} R_{ij}-b_{ij}^k & \text{if } R_{ij} > b_{ij}^k \\ R_{ij} & \text{if } R_{ij} < b_{ij}^k \end{pmatrix}$$

$$B_{ij}=B_{ij}+b_{ij}^k \text{ for all(i,j)}$$

위의 알고리즘이 동작하기 위해서 모든 노드는 현재 링크 상에서 각각의 작업 경로가 차지하고 있는 대역, 각각의 복구 경로가 차지하고 있는 대역, 어떤 작업 경로에 대한 복구 경로인지, 그리고 각 링크에서 작업 경로와 복구 경로에 할당하고 남아 있는 대역의 정보를 필요로 한다.

## IV 결과 분석

### 1. 디스조인트 경로 계산 알고리즘과 결과

본 논문에서 제시한 경로 설정 알고리즘을 수행하기 위해서는 디스조인트 경로 계산 알고리즘이 필요하다. 작업 경로와 이를 대체할 수 있는 복구 경로는 작업 경로의 대역을 보장할 수 있는 복구 경로로서 작업 경로와 복구 경로는 시작 노드에서 종점 노드까지 두 경로 간에는 링크를 전혀 공유하지 않는 경로가 된다.

본 논문에서는 이러한 디스조인트 경로 계산을 위한 알고리즘으로 두 가지 방법을 고려하였다. 첫째 방법은 작업 경로를 가장 짧은 경로로 설정하고 디스조인트를 위해 작업 경로 상의 링크를 제외한 링크 중에서 가장 짧은 경로를 복구 경로로 설정하는 방법이다.

즉, 먼저 가장 짧은 경로를 설정하고 그 경로를 작업 경로로 한다. 다음에 작업 경로를 임시로 지우고 (pruning), 다시 가장 짧은 경로를 설정하여 그 경로를 복구 경로로 사용한다.

두 번째 방법은 [8]에서 제안한 것과 같은 작업 경로와 복구 경로를 한하여 홉 수가 가장 작은 두 경로를 구하는 것이다. 먼저 가장 짧은 경로를 설정한다. 다음에 설정된 경로의 방향을 바꾼 뒤, 다시 가장 짧은 경로를 구하고 먼저 구한 경로와 다음에 구한 경로가 겹치는 링크들을 모두 임시로 지운다. 그런 다음 가장 짧은 경로를 구하여 그 경로를 작업 경로로 하고, 다음에 또 다시 최단 경로를 구하여 그 경로를 복구 경로로 사용한다.

본 논문에서는 그림 2와 같은 실제 망의 토폴로지에 대해서 시뮬레이션을 수행하였다. 이 망은 27개의 노드와 45개의 양방향 링크로 구성되어 있다[9]. 각 노드들은 인접 노드와 연결된 링크에 대한 정보를 가지고 있다. 각 링크는 연결된 두 노드에 대한 정보, 남은 대역 그리고 복구 경로가 보호하는 원래의 작업 경로의 정보와 대역폭, 그리고 복구 경로의 대역에 대한 정보를 가지고 있다. 그리고 경로 설정을 위한 정보로서 요구대역, 지나가는 노드들에 대한 정보를 기록하고 있다. 경로를 계산할 때의 링크 비용은 현재 가용 대역 크기의 역수로 하였다.

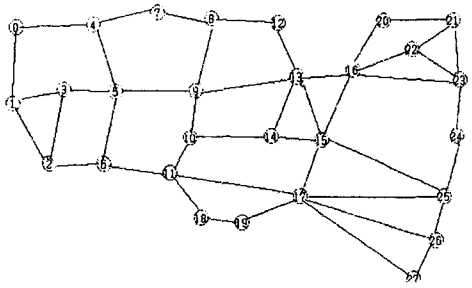

그림 2: 망 토폴로지
Fig 2: Network topology

본 논문에서는 두 가지 알고리즘의 복잡도를 비교하기 위하여 그림2의 토폴로지 상에서 작업 경로와 복구 경로를 계산하고 각 경로를 구성하는 홉 수를 비교해 보았다. 그림3은 이 결과를 보여주고 있다. 이 결과를 보면 두 방법이 모두 작업 경로의 평균 홉 수가 3이고 복구 경로의 평균 홉 수가 4로 거의 차이가 없는 것을 알 수 있다. 따라서 두 경로의 홉 수를 고려한다 하더라도 두 방법 간에 대역의 소비를 구하는데 미치는 영향은 거의 없다는 것을 알 수 있다. 본 논문에서는 각 경로의 대역 소비와 거부 수를 계산할 때 첫 번째 방법으로 디스조인트 경로를 계산하였다.

## 2. 대역 소비와 거부

시뮬레이션은 다음 두 가지 경우에 대해서 수행하였다. 첫째로 대역 소비 정도를 보기 위해서 각 링크에 대한 대역을 무한대로 하고 작업 경로에 대한 요구를 발생시켰다. 대역이 무한대이기 때문에 경로 설정 거부는 일어나지 않는다.

그림4는 그림2의 토폴로지에 대해서 작업 경로와 복구 경로의 총 요구 대역과 복구 경로가 없는 경우 작업 경로의 요구 대역을 경로 설정 요구(path request)의 수에 대해서 보여 주고 있다. 이 실험에서는 각 경로 설정에 대해서 1에서 10 크기의 랜덤한 크기의 대역을 요구하였다. 각 링크의 대역은 100에서 300까지 임의로 할당하였다.

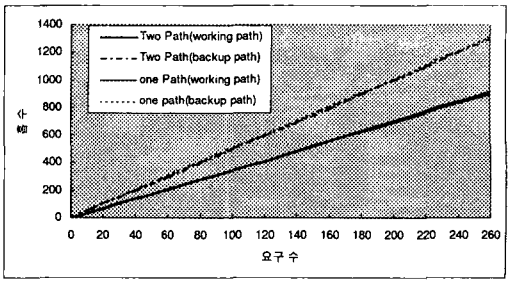


그림 3: 경로 설정 요구 수 vs. 홉 수
Fig3: path setup request vs. hop numbers

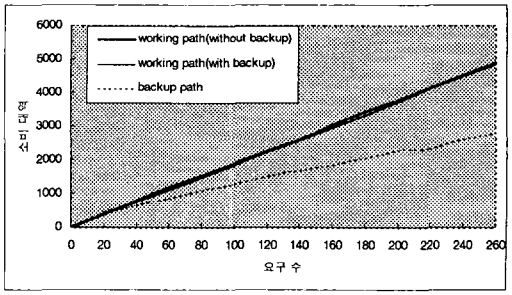


그림 4: 경로 설정 요구 수 vs. 대역 소비(링크대역은 무한)
Fig3: path setup request vs. bandwidth consumption (infinite bandwidth)

그림4에서 보는 바와 같이 200번 설정 요구 시에 3880의 작업 경로 대역과 2006의 복구 경로 대역이 요구된다. 시뮬레이션의 결과 작업 경로와 복구 경로의 홉 수의 비는 그림3에서 보는 바와 같이 약 3:4 정도이다. 만약 공유를 하지 않는다면 복구 경로의 필요 대역은 작업 경로의 필요 대역과 같아야 한다. 따라서 홉 수를 고려 할 때 복구 경로의 대역 소비는 작업 경로보다 많은 대역을 소비해야 한다. 그러나 공유를 할 경우 복구 경로의 대역 소비를 작업 경로의 대역 소비에 비해서 감소할 수 있으며 그림4는 감소 정도가 어느 정도 인지를 보여주고 있다. 따라서 이 알고리즘은 공유 방식에 의해 복구 경로를 설정할 경우 대역 소비가 어느 정도인지를 판단할 수 있도록 하며 이 결과는 망의 설계에 있어서 적절한 링크의 대역을 결정하는데 활용할 수 있다.

두 번째로 경로 설정 요구를 할 때 현재 링크의 가용 대역이 요구 대역 보다 작을 경우 설정 요구를 거부하게 된다. 대역 요구의 거부는 작업 경로의 요구와 복구 경로의 요구, 두 요구 모두 해당한다. 이러한 경로 설정 요구에 대한 거부의 효과를 보기 위해서 망의 각 링크에 유한 대역을 할당하였다. 그림5는 각 링크의 대역을 200으로 균등하게 할당할 경우의 거부의 수를 보여주고 있으며, 그림 6은 링크의 대역을 100에서 300 사이의 값으로 상이하게 할당한 경우의 결과를 보여주고 있다. 두 경우에 대해서 각 경로 설정 요구는 1에서 10 사이의 랜덤한 크기의 대역을 요구하도록 하였다.

각 링크에 일정한 크기의 대역을 할당한 그림 5의 경우 거부의 발생은 그림6의 경우에 비해서 빨리 발생하는 것을 볼 수 있다. 그림6에서 보는 바와 같이 복구 경로 설정의 경우 경로 설정 요구의 수가 350번이 되면 설정 거부가 발생하며 그림5의 300번 보다 평균 50번 설정 요구 후에 거부가 발생하는 것을 볼 수 있다.

그림5와 6의 결과를 볼 때 링크 대역의 분배 방식에 따라 거부가 영향을 받는 것을 알 수 있다. 이와 함께 망의 토폴로지도 거부 효과에 영향을 주고 있다. 그림2의 토폴로지에서는 경로 4-7-8, 5-9, 6-11의 세 가지 경로가 중간 다리 역할을 하기 때문에 어느 한 링크가 상대적으로 큰 대역을 갖고 있다면 우회 경로를 찾게 되어 거부 효과를 줄일 수 있기 때문이다.

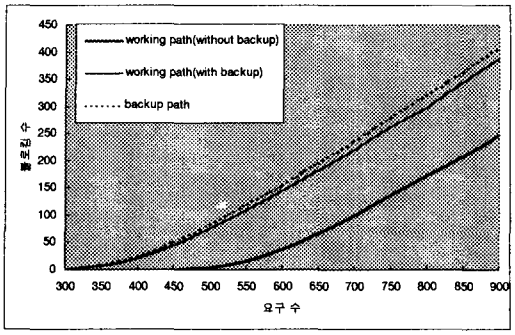


그림 5: 설정 요구 수 vs. 거부 수 (링크의 대역은 일정)
Fig5: Path setup request vs. blocking (uniform bandwidths)

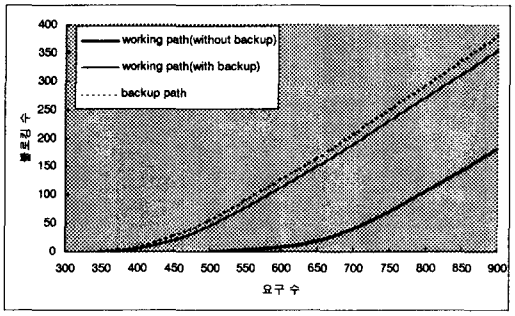


그림 6: 설정 요구 수 vs. 거부 수 (링크의 대역은 가변)
Fig5: Path setup request vs. blocking (nonuniform bandwidths)

## V 결 론

본 논문에서는 망에 발생하더라도 서비스가 지속적으로 제공될 수 있도록 하기 위해서 대역을 보장하면서 작업 경로와 복구 경로를 설정하는 알고리즘을 제시하고 시뮬레이션을 통해서 그 결과를 구해 보았다. 먼저 두 가지 디스조인트 경로 계산 알고리즘을 비교하였다. 그리고 시뮬레이션을 통해서 제안한 알고리즘의 대역 소비와 거부 효과를 분석하였다.

이 결과를 통해서 제시된 알고리즘이 작업 경로가 동일 링크를 지나가지않는 복구 경로들 간에는 대역을 공유할 경우 어느 정도 대역의 소비를 감소할 수 있는지를 보여 주었다. 이 알고리즘은 망의 신뢰성을 높이기 위해서 망 복구 경로를 설정하는 경우 링크 대역의 크기 예측, 그리고 설정 요구 시 거부의 효과를 고려한 망 토폴로지의 구성과 링크 대역의 할당에 효과적으로 활용될 수 있을 것이다.

이 알고리즘에서 모든 노드는 현재 링크 상에서 각각의 작업 경로가 차지하고 있는 대역, 각각의 복구 경로가 차지하고 있는 대역, 어떤 작업 경로에 대한 복구 경로인지에 대한 정보를 필요로 한다. 따라서 노드가 증가할수록 확장성의 문제가 발생할 수 있다. 이런 점에서 이 알고리즘은 각 노드가 이런 정보를 갖는 것보다는 한 노드가 모든 정보를 갖고 관리하는 경우에 적절하게 사용될 수 있을 것이다.

## 참고문헌


[1] D. O. Awduche et al," "Multi-Protocol Lambda Switching: Combining MPLS Traffic Engineering Control with Optical Crossconnect," Internet Draft <draft-awduche-mpls-te-optical-02.txt>, July 2000[1]

[2] V. Sharma et al., "Framework for MPLS-based Recovery," Internet Draft <draft-ietf-mpls-recovery-frmwrk-03.txt>, July 2001

[3] B. T. Doshi et al, "Optical Network Design and Restoration," Bell Lab Technical Journal, January-March 1999

[4] M. Kodialam and T.V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," INFOCOM 2000

[5] S. Kini et al, "Shared backup Label Switched Path restoration," Internet Draft <draft-kini-restoration-shared-backup-00.txt>, November 2000

[6] 이황규, 홍석원, "대역을 보장하는 복구 경로 설정 알고리즘의 분석,"

[5] D. O. Awduche et al, "A Framework for Internet Traffic Engineering," Internet Draft <draft-ietf-tewg-framework-02.txt>," July2000

[8] J.W. Suurballe and R.E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," Networks(14), 1984

[9] K. Murakami, "Optimal Capacity and Flow Assignment for Self-Healing ATM Networks Based on Line and End-to-End Restoration," IEEE/ACM Trans. On Networking 6(2), April 1998


## 저자소개


이황규(Hwang-Kyu Lee)

2000년 2월 명지대학교 정보통신공학과(공학 학사)

2001년~현재 명지대학교 정보통신공학과(석사과정)

홍석원(Sug-Won Hong)

1972.2 서울대학교 물리학과(학사)

1988.7 North Carolina State University 전산학(석사)

1992.7 North Carolina State University 전산학(박사)

1993.2-1995.2 전자통신연구원 광대역 통신망 연구부 선임연구원

1995-3-현재 명지대학교 전자정보통신공학부 부교수

※관심 분야 : 망 구조 및 프로토콜 설계, 성능 분석