다중경로 패킷 전달환경에서의 안정적인 부하제어 기법
Stable Load Control in Multipath Packet Forwarding

박일규(朴一圭, Ilkyu Park),
김종성(金鍾成, Jong-Sung Kim)
한국전자통신연구원 (Electronics and Telecommunications Research Institute) *

이영석(李泳錫, Youngseok Lee),
최양희(崔陽熙, Yanghee Choi)
서울대학교 (Seoul National University)**

연구세부분야: 정보통신
키워드 : 부하제어, 다중경로 패킷 전달, MPLS, 트래픽 엔지니어링
*주소 : 대전시 유성구 가정동 161번지 한국전자통신연구원 VR센터
우편번호: 305-350
전화 : 042-860-6611
FAX : 042-860-1051
E-Mail : xiao@etri.re.kr

다중경로 패킷 전달환경에서의 안정적인 부하제어 기법
Stable Load Control in Multipath Packet Forwarding

요약

MPLS와 트래픽 엔지니어링 기술의 발전으로 다중경로 패킷 전달 및 동적인 부하 제어가 가능하게 되었다. 동적인 부하 제어는 네트웍의 상태를 고려하여 경로간의 부하를 조절함으로써 네트웍의 효율성을 높일 수 있으나, 상태 정보의 시간차이에 의해 불안정한 상태에 이르기 쉽다. 본 논문에서는 부하의 할당을 안정적으로 변화시키면서도 효율적인 부하 제어를 하는 기법을 제안하고자 한다. 제안하는 기법에서는 네트웍 상태 정보에 예측기법을 도입하여 시간차이에 의한 오류를 줄이고 진동을 막는 효과가 있다.

Abstract

With the invention of MPLS and the improvement in traffic engineering, multipath packet forwarding and dynamic load control has become a reality. A dynamic load control, while it can improve network efficiency by controlling loads between paths according to the network state, can lead to unstable and oscillating state because of the staleness of the state information. In this paper, we propose a efficient load control scheme which remains stable. The proposed scheme introduces prediction to reduce the staleness of state message and to prevent oscillation.

1. 서론

네트웍에 혼잡(congestion)이 발생하는 경우는 크게 두 가지 경우가 있는데, 하나는 망의 자원이 부족하여 입력 부하를 처리할 수 없는 경우이며, 두 번째의 경우는 트래픽 스트림이 네트웍의 자원에 비효율적으로 배치되어 네트웍의 일부는 사용되지 않고 일부는 과도하게 사용되는 경우이다. 첫 번째 혼잡은 고전적인 혼잡제어(congestion control)를 사용하여 해결할 수 있다. 혼잡제어는 트래픽율를 제한하고 흐름제어를 사용하여 망에 과도한 부하가 들어오는 것을 막아준다. 두 번째 경우에도 혼잡제어가 사용될 수 있지만, 이 방식은 입력되는 부하를 줄이는 방식이므로 네트웍을 비효율적으로 이용하게 된다.

두 번째 경우의 혼잡을 해결하는 데는 트래픽 엔지니어링이 필요하다. 트래픽 엔지니어링은 입력된 부하를 네트웍의 자원에 효율적으로 매핑하는 문제를 다룬다. 과거에는 트래픽 엔지니어링을 위한 네트웍의 지원이 부족하였는데, 이는 인터넷의 내부 게이트웨이 프로토콜들이 트래픽 엔지니어링을 위한 충분한 제어 기능을 제공하지 않고, 인자값의 변화를 이용하여 라우팅을 제어하는

기법은 네트워크 전체에 영향을 주어 매우 거친 정도의 조작만이 가능했던 것이다. 그러나, 최근에 도입된 MPLS(Multiprotocol Label Switching)는 패킷 전달 속도를 빠르게 하였을 뿐만 아니라, 트래픽 엔지니어링에 필요한 기반 기능들을 제공하고 있다. MPLS는 트래픽 흐름을 명시적 LSP(Label Switched Path)로 설정할 수 있어 토폴로지 기반의 라우팅 프로토콜의 제한을 받지 않는다. 또한 LSP는 관리가 간단하며 트래픽 트렁크를 하나의 LSP로 매핑할 수 있다. LSP에는 특성을 설정할 수 있어 경로의 배치를 조절할 수 있고, 이는 제한 기반 라우팅 프레임워크을 쉽게 결합할 수 있게 해 준다.

트래픽 엔지니어링을 도입하기 위해서는 MPLS뿐만 아니라, 각 FEC(Forwarding Equivalence Class)에 사용할 LSP를 찾는 알고리즘이 필요하고, 실제의 트래픽 흐름을 LSP로 할당하는 방법이 필요하다. 또한 망의 상태를 고려하는 동적인 트래픽 제어를 하기 위해서는 링크 상태를 교환하는 프로토콜이 도입되어 가용 대역폭이나 트래픽 특성 등의 링크 상태를 교환할 수 있어야 한다.

네트워크의 상태를 고려하는 트래픽 엔지니어링은, 네트워크의 트래픽 정보들을 기록하고 이를 바탕으로 오프라인으로 최적화를 하여 네트워크 전체를 다시 구성하는 장기간 방식이 있다. 이 경우에는 트래픽 데이터가 하루 또는 일 주일 단위로 모여 최적화에 사용되게 된다. 데이터를 사용하여 필요한 LSP들을 구성하고, 요구조건을 만족하는 LSP배치가 이루어지게 된다. 또 하나의 방식은 현재의 네트워크 상황을 고려하여 트래픽 흐름을 배치하는 온라인방식이다. 각 라우터들은 자신의 상태를 다른 라우터와 교환하여 네트워크의 상태를 알고 이에 따라 트래픽 흐름을 조절하게 된다. 이 상태정보의 교환은 링크상태 라우팅 프로토콜에서 사용되는 링크상태 알림을 이용하여 전달할 수 있다. 각 라우터는 다른 링크의 상태 정보, 즉 가용 대역폭이나 부하 특성 등을 얻어 혼잡을 최소화하는 방식으로 트래픽을 배치하게 된다.

그러나, 이 경우에는 알고리즘의 분산성과 동적인 성질로 인해 불안정한 결과를 만들기 쉽다. 라우터는 주기적으로 얻는 상태정보를 통해 네트워크의 상태를 얻기 때문에, 실제의 상태와 전달된 상태에 차이가 생기고, 이로 인해 부하 제어가 불안정해진다. 본 논문에서는 이 문제점, 즉 네트워크의 상태를 고려하는 온라인 부하 제어 방식에서의 불안정성에 초점을 맞추고, 이를 해결할 수 있는 간단하면서도 효율적인 알고리즘을 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2절에서는 기존의 연구 내용과 본 논문에서 해결하고자 하는 문제점을 설명한다. 3절에서는 본 논문의 아이디어와 구체적인 알고리즘을 기술하며, 4절에서는 제안된 알고리즘의 모의실험 결과와 분석을 한다. 마지막으로 5장에서 결론을 맺는다.

2. 다중경로 패킷 전달과 동적인 부하제어

2.1 부하제어 방식

이 절에서는 다중경로 패킷 전달환경 하에서 동적 부하제어를 하는 방법을 기술한다. 먼저, 사용할 경로(LSP)들을 만드는 알고리즘이 필요하다. 경로를 만드는 알고리즘은 오랫동안 연구가 되어 왔으며, 많은 알고리즘들이 제시되어 있다[1,2]. 이렇게 만들어진 경로들을 동시에 사용하려면 경로의 정보를 유지해야 하며, 패킷 전송시 여러 경로를 통해 전송하는 방식이 제공되어야 한다. 경로의 정보를 유지하는 것은 ECMP(Equal Cost MultiPath)나 MPLS를 이용하여 가능하다. ECMP에서는 경로의 선택이 같은 경로값을 가진 경로로 제한된다. 본 논문에서는 MPLS를 이용하여 명시적 패킷 전달을 하고 이 경로가 LSP(Label Switched Path)라고 가정한다. 또한 입력된 트래픽을 이들 LSP중 하나로 매핑하는 방법이 필요하다. 이 매핑은 흐름단위에서 행해져야 하는데, 패킷 단위의 매핑은 순서가 바뀐 패킷을 만들어 TCP의 성능을 감소시키기 때문이다. 그리고, 이 방법은 자유롭게 매핑을 바꿀 수 있어야 LSP간에 부하를 옮겨 부하 제어를 할 수 있다. 이 매핑 방법은 해싱기반의 패킷 매핑[3]을 사용할 수 있다.

부하제어를 위한 또 하나의 요구조건은 링크상태정보 교환 프로토콜이다. 각 경로의 상태에 따라 동적으로 부하 제어를 하기 위해서 라우터들은 각 경로 상에 있는 링크들의 상태를 알아야 한다. 이 정보는 OSPF[4]와 같은 링크상태 라우팅 프로토콜에 QoS 확장[5]을 적용하여 교환할 수 있다. 라우터간에 교환되는 링크상태 정보는 각 링크에 걸린 부하와 그 링크의 여유 대역폭을 포함하거나, 링크의 부하와 대역폭을 포함해야 한다. 이 값은 또한 전송률과 링크의 손실률로부터 계산할 수도 있다.

부하제어 문제는 입력 부하를 가능한 LSP들에 할당시키는 문제이다. 이 할당방법은 망의 자원이 어떻게 사용될 지 결정하며, 나아가 망 전체의 처리량을 결정하게 된다. 망의 부하는, 혼잡이 생기는 링크와 사용되지 않는 링크가 최소화 되도록 할당되어야 한다. 각 라우터는 링크상태 메시지에 의해 얻어진 링크들의 상태를 바탕으로 입력 부하를 LSP들로 할당한다. 링크 상태 정보를 바탕으로 각 LSP의 가용 대역폭을 계산하게 되고, 대역폭의 크기에 비례하여 입력 부하를 그 LSP에 할당한다. 각 패킷을 전달하는 경우에는, 해싱을 사용하여 해쉬값에 해당하는 LSP로 패킷을 전달하게 된다. 해싱을 사용하므로 같은 흐름에 속한 패킷들은 같은 LSP로 할당되어 순서가 바뀐 패킷이 생기는 것을 방지할 수 있고, 해쉬 table을 조절하여 쉽게 LSP로 할당되는 부하의 비율을 조절할 수 있다.

2.2 동적 부하 제어의 불안정성

앞 절에서 설명한 방식에서, 라우터가 LSP에 할당하는 부하의 양은 링크상태 메시지를 통해 얻어진 링크들의 상태에서 결정된다. 그런데, 링크상태 메시지가 생성된 뒤에도 각 링크의 상태는

계속 변하고, 이 변화는 다음 링크상태 메시지가 생성될 때까지 다른 라우터들에게 알려지지 않아 각 라우터가 알고 있는 링크의 상태와 실제의 링크의 상태는 차이가 생기게 된다. 게다가, 부하 제어는 링크의 상태 정보를 이용하여 다시 링크의 상태에 영향을 주기 때문에 이 차이가 더욱 심해지고, 메시지를 자주 전송하더라도 상태의 차이를 줄일 수가 없게 된다.

앞 절에서 설명한 방식에서는 경로의 여유 대역폭에 비례하는 양의 부하를 그 경로에 할당하므로, 부하가 적은 링크가 발견되면 라우터들은 그 링크에 할당된 부하를 증가시키게 된다. 링크를 사용하는 라우터들의 집합을 알지 못하므로 각 라우터는 독립적으로 부하를 할당하게 되고, 결과로 할당된 부하는 링크를 혼잡상태로 만들게 된다. 다음 링크상태 메시지가 전송되면 라우터들은 이 링크가 혼잡상태라는 것을 알게 되고, 역시 독립적으로 링크의 부하를 줄이게 된다. 따라서 이 링크는 다시 여유가 많은 링크가 되고, 다음 번 메시지 전송 후에는 혼잡이 발생하는 상태가 계속 반복된다[6].

3. 예측기반 부하 제어 알고리즘

3.1 예측기법을 사용하는 부하제어

2.2절에서 설명한 동적 부하 제어의 불안정성은 링크상태 메시지로 교환된 정보와 현재의 링크상태 사이의 오차에서 기인한 것이다. 따라서, 불안정성을 줄이고 진동을 막기 위해서는 현재의 망의 상태를 예측하는 방법을 사용해야 한다. 본 논문에서는 링크의 부하가 증가하는 속도를 이용하여 예측을 한다. 각 라우터가 독립적으로 부하 제어를 할 때에 부하가 증가하는 속도는 해당 링크를 사용하는 라우터의 수에 비례하게 되고, 따라서 이전에 교환된 링크의 상태와 현재의 망의 상태의 차이도 이에 비례한다. 그러므로, 이전에 링크의 부하가 증가하는 속도를 관측하면 현재의 링크 상태정보를 예측할 수 있다. 부하의 증가 속도는 일시적인 변화의 영향을 피하기 위해 α값이 낮은 이동 평균값을 사용하였다. 링크 i의 부하 증가속도 s(i)를 유지하는 식은 알고리즘 1에 기술하였다. 여기서, PL은 이전 부하, CL은 현재의 부하를 가리킨다.

알고리즘 1 : 링크상태갱신(i, Load)

```
if (PL(i) < Threshold) and (PL(i) < CL(i)) then
        s(i) := s(i) * (1 - α) + (CL(i) - PL(i))/UpdatePeriod * α
end if
PL(i) := CL(i)
```

트래픽 엔지니어링은 일반적으로 대역폭이 높고 트래픽이 합쳐지는 코어 쪽에서 수행되기 때문에 본 논문에서는 제안하는 알고리즘이 코어 네트워크에서 사용된다고 가정하였다. 또한, 코어 네트워크에서는 많은 종류의 트래픽이 합쳐져 트래픽율의 변화가 적은 균일 비트율과 비슷한 트래픽 특성을 얻을 수 있다. 링크의 부하가 증가하는 속도는 입력 트래픽에 의해 변화가 생기게 되지만, 코어 네트워크에서는 트래픽의 변화가 적어 이 영향을 무시할 수 있다. 또한, 트래픽율를 긴 주기의

평균을 사용하면 일시적인 변화가 평균에 영향을 주지 못하기 때문에 충분히 변화가 없는 트래픽을 얻을 수 있다.

예측을 하기 위해서는 각 라우터는 하나의 링크에 부하를 증가시킬 것인지 감소시킬 것인지 판단하는 공통된 기준이 필요하다. 이를 위해 우리는 공통의 문턱값 T를 도입하고, 링크의 부하가

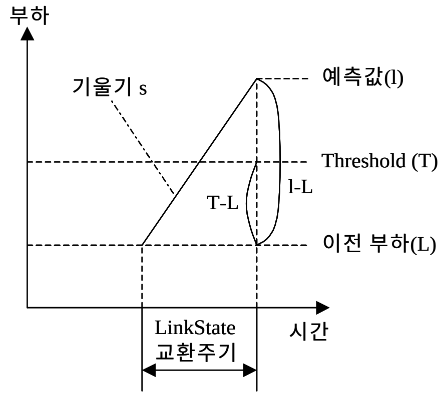


그림 1 : 부하 예측

그 링크의 대역폭의 T 퍼센트를 넘으면 링크의 부하를 줄여야 하는 상황으로 정하여 부하를 줄이도록 하였다. 링크의 부하가 링크 대역폭의 T퍼센트보다 작을 때에는 그 링크는 부하를 할당받을 수 있는 상황으로 정하여 각 라우터들이 링크에 트래픽을 할당하게 된다. 이 기준을 이용하면 라우터들은 링크의 부하가 증가할 지 감소할 지 알 수 있어 경로의 부하를 예상할 수 있다. 문턱값 T는 망 관리자가 정하는 값으로서, 낮은 T값을 사용하면 부하를 할당할 수 있는 링크가 줄어들어 부하 제어가 비효율적이 되며 높은 T값을 사용하면 여유 대역폭이 적어 패킷이 손실될 확률이 높아진다. 문턱값 T와 기울기 s를 이용하면 예측값은 알고리즘 2 와 같다.

알고리즘2 : LoadEstimation(i)

```
if CL(i) < T then
    EL(i) := CL(i) + s(i)*UpdatePeriod
else
    EL(i) := CL(i)
end if
```

예측을 한 후에는 부하 제어에 예측결과를 적용해야 한다. 링크의 상태를 혼잡하지 않은 상태로 유지하기 위해, 현재의 링크의 부하에 비례하여 부하를 줄이는 방식으로 부하 제어를 한다. 예측된 부하 정도를 l(=현재부하 + 링크상태교환주기*s)이라 하고, 문턱값을 T, 마지막 링크상태 교환에서 얻어진 링크의 부하를 L이라 하면, 초과된 부하의 양은 l-T가 되고, 부하 제어로 인한 부하 증가는 l-L 이 된다(그림 1). 따라서, 다음 링크 상태 교환이 일어나는 시기의 부하를 문턱값로 유지하기 위해서는 부하의 증가를 (T-L)/(l-L) 만큼 늦출 필요가 있다.

예측된 부하를 이용하여, 경로의 부하 비율을 조절함으로써 부하 제어를 수행한다. 각 목적지로 향하는 트래픽 양 중에서, 특정한 경로에 할당할 트래픽의 비율을 부하 비율로 정의한다. 링크의 대역폭으로부터 각 LSP의 대역폭이 계산되면. 부하의 배분은 문턱값보다 부하 수준 (물리적 대역폭에 대한 트래픽 양의 비율)이 높은 LSP들에서 다른 LSP로 부하 비율을 옮김으로써 수행된다. T보다 높은 링크에서는 부하를 낮춰 T부근으로 만들어야 하므로, 목표하는 부하 비율은 (현재의 부하비율)*T/(현재의부하수준) 이 되고, 이 값과 현재의 부하비율의 차이만큼이 옮기는 양이 된다. 만약 다른 LSP들로 부하 비율을 옮길 여유가 없다면 현재의 상태를 유지한다.

알고리즘 3 AssignWeight(d)

```
G := 목적지가 d 인 LSP의 집합
for all LSP in G do
   RB(LSP) := LSP상의 링크들의 가용 대역폭의 최소값, EstimateLoad(i)사용
   BW(LSP) := 병목 링크의 물리적 대역폭
  PRA(LSP) := LSP의 이전 부하 비율
  문턱값보다 부하율이 높은 경우 표시
end for
RES_BW_RATIO_SUM := (문턱값보다 부하율이 낮은 경로들의 가용 대역폭 합)
        / (이전 입력 부하)
MOVE_SUM := 0
H := G의 LSP중 1-RB(LSP)/BW(LSP) > 문턱값 인 것의 집합
for all LSP in H do
    NEXT_RATIO = PRA(LSP)*(BW(LSP)*문턱값)/(BW(LSP)-RB(LSP))
    MOVE_RATIO = PRA(LSP)-NEXT_RATIO
    if MOVE_RATIO + MOVE_SUM < RES_RATIO_SUM then
       SetRatio(LSP, NEXT_RATIO)
       MOVE_SUM = MOVE_SUM + MOVE_RATIO
    end if
end for
I := G - H
PartialSum := 0
for all LSP in I do
    if (BW(LPS)-RB(LSP))/BW(LSP) > 문턱값 then
            RB(LSP) := RB(LSP) * (RB(LSP) - CL(LSP))/(문턱값 -
CL(LSP)/BW(LSP))
       PartialSum := PartialSum + RB(LSP)
    end if
end for
for all LSP in I do
    SetRatio(LSP, PRA(LSP) + MOVE_SUM*RB(LSP)/PartialSum)
end for
```

부하 수준이 T보다 낮은 LSP들에는 위에서 줄인 만큼의 부하가 할당되어야 한다. 이 때, 예측된 부하 수준 역시 T보다 낮을 경우는 가용 대역폭에 비례하는 양의 부하 비율을 할당할 수 있지만, 예측된 부하 수준이 T이상일 경우에는 할당되는 양을 줄여야 한다. 위에서 설명한 바와 같이 늦추는 비율은 (T-L)/(1-L)이므로, 현재의 가용 대역폭에 이 값을 곱하고, 결과 값에 비례하는 양으로 부하 비율을 할당하게 된다. 최종 알고리즘은 알고리즘 3 에 기술하였다.

4. 모의실험

4.1 실험방법

본 논문에서는 이벤트 방식 망 모의실험 프로그램인 NS-2[7]를 이용하였다. NS-2는 패킷방식의 망에 대해 모의실험을 지원하며, 인터넷의 모의실험에 필요한 TCP, 라우팅 프로토콜, 트래픽 생성 등의 포괄적인 기능을 갖추고 있다. 이번 연구를 위해서 우리는 NS-2에 링크 상태 라우팅 프로토콜 모듈을 추가하였으며, 제안한 예측기반 부하제어 알고리즘을 구현하였다. 다중경로 패킷 전달을 위해서는 NS-2의 MPLS 모듈을 사용하였다.

4.1.1 Topology

제안한 알고리즘은 코어 망을 대상으로 하기 때문에 모의실험으로는 간단한 토폴로지를 사용하였다. 그림 2에 이 토폴로지가 나와 있다.

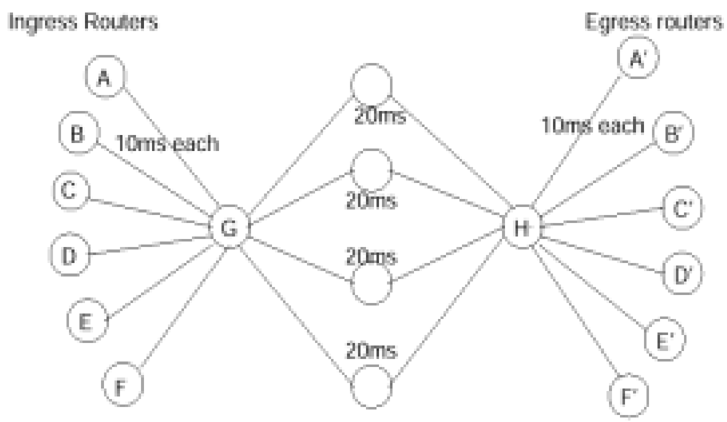


그림 2 모의실험 토폴로지

망의 중심부에는 두 개의 라우터를 두고, 라우터 사이에 여러 개의 경로를 만들었다. 코어 망에서 두 라우터 사이의 중복이 없는 경로의 개수는 일반적으로 2개에서 4개 정도이다. 이 모의실험에서는 부하제어의 효과를 나타내기 위해 4개의 중복없는 경로를 만들었다. 한 쪽 라우터(G)에 여러 개의 입구 라우터가 연결되고, 출구라우터는 다른 쪽 라우터(H)에 연결하였다. 각 입구-출구 라

우터 쌍은 코어 망의 다중 경로를 사용하며, 독립적으로 부하 제어를 수행한다. 입구-출구 라우터 쌍이 많아질수록 부하 제어는 힘들어진다. 여기에서는 여섯 개의 입구-출구 라우터 쌍을 배치하였다. 각 링크의 대역폭은 모의실험에 따라 설정하였으며, 코어 망의 링크는 ATM등 고속 망을 사용하므로 링크의 지연은 10ms로 설정하였다.

다중경로 패킷 전달을 하기 위해서 MPLS 모듈을 사용하였다. MPLS모듈을 사용하면 각 라우터 사이에 명시적인 LSP를 만들 수 있다. 이 실험에서는 각 라우터 쌍마다 네 개의 명시적 LSP를 만들어 총 24개의 LSP가 만들어진다. G와 H사이의 링크들에는 각각 6개의 경로가 사용된다. 라우터 쌍들 사이의 경로들이 중복되는 링크를 많이 사용할수록 라우터들은 서로 영향을 주게 되고, 기초적인 부하 제어방식은 비효율적이 된다. 각 라우터 쌍들은 다른 라우터 쌍이나 다른 라우터의 경로에 대해 정보를 가지고 있지 않다. 이 토폴로지에 대해 생성되는 논리적인 토폴로지는 그림 3에 도시되어 있다.

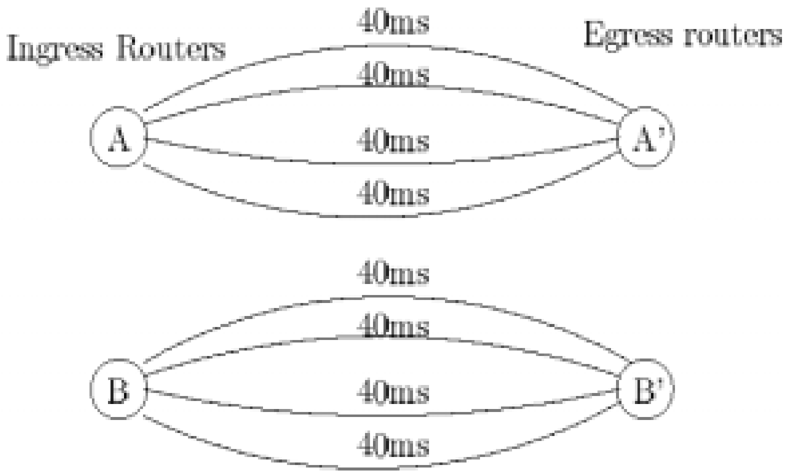


그림 3 논리적 토폴로지

4.1.2 입력 트래픽

알고리즘의 성능을 평가하기 위하여 여러 가지의 입력 트래픽에 대해 모의실험을 행하였다. 첫 번째 종류의 실험에서는 CBR(균일 비트율) 트래픽을, 두 번째에는 CBR에 중간에 트래픽율의 변화가 있는 트래픽을, 세 번째에는 NLANR 네트워 헤더 기록[8]을 이용한 실험을 하였다. CBR 트래픽의 부하는 6Mbps(총 대역폭의 100%), 5.2Mbps(87%), 4.2Mbps(70%)를 사용하였으며, 변화가 있는 두 번째 모의실험에서는 4.8Mbps에서 3.6Mbps로 트래픽율가 변화한 후 다시 4.8Mbps로 돌아가는 경우와, 3.6Mbps에서 시작하여 4.8Mbps로 늘어난 후 다시 3.6Mbps로 변하는 트래픽을 사용하였다. 트래픽 트레이스를 사용하는 경우는, 트래픽의 양을 조절할 수 없으므로 네트워

의 대역폭을 변화시켜 네트웍의 사용율이 100%, 90%, 80%가 되는 경우의 모의실험을 수행하였다.

4.1.3 모의실험 인자

모의실험 결과에 영향을 주는 인자가 여러 가지 있다. 이번 모의실험에서는 문턱값을 80%로 사용하였다. 이 문턱값은 부하 제어의 목표가 되는 부하 수준이 된다. 이 값이 높으면 가드 대역폭이 충분하지 않아 링크가 빈번히 혼잡되게 된다. 만약 너무 낮을 경우에는 모든 링크가 혼잡된 것으로 판단되어 부하 제어가 이루어지지 않는다.

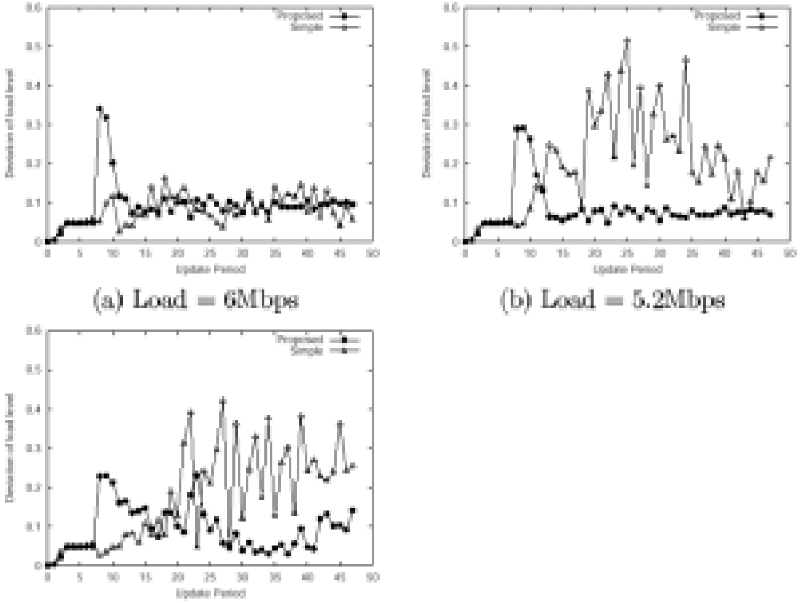
트래픽 증가율의 이동 평균을 계산할 때 사용하는 알파값은 낮은 값으로 하여야 잘못된 예측으로 인한 진동이 발생하지 않는다. 그러나, 이 값이 낮으면 이동 평균값이 수렴하는 시간이 오래 걸린다. 그러나, 우리는 부하 제어의 불안정성에 초점을 두고 있으므로 이 속도는 큰 영향을 미치지 않는다. 사용한 알파값은 0.3이다.

4.2 모의실험 결과

4.2.1 균일 비트율 트래픽

예측을 사용하지 않은 경우와 예측을 사용한 결과의 비교가 그림 4에 나와 있다. 그래프에서 하나의 시간 단위는 링크상태 주기에 대응된다. 그래프의 세로축은 링크간의 부하 수준의 표준편차이다. 표준편차가 낮을수록 부하가 고루 분포된 것을 의미한다. 표준편차는 모의실험 토폴로지 중 가운데의 4개의 링크 사이에서 계산하였다.

부하가 높을 경우 (그림 4 (a))에는 두 방법 모두 낮은 편차를 보인다. 이 경우에는 가용 대역폭이 거의 없어 부하 제어가 동작하지 않아 경로간 부하의 이동이 없는 것을 알 수 있다. 부하가 보통인 경우 (그림 4 (b))에는 예측을 사용하지 않은 알고리즘은 심한 진동과 높은 표준편차를 나타내고 있다. 예측을 사용한 결과는 안정적이면서 낮은 표준편차 수준이 계속 유지됨을 알 수 있다. 부하가 더욱 낮은 경우에는 (그림 4 (c)) 예측을 사용하지 않은 경우는 진동과 함께 표준편차가 계속 증가함을 알 수 있다. 예측을 사용한 경우, 초기에는 예측의 오차로 인해 큰 표준편차를 보이지만 곧 편차가 감소하는 결과를 보이고 있다. 세 그래프에서, 예측을 사용한 경우에 진동을 막을 수 있고 균일한 부하 제어가 일어나는 결과를 확인할 수 있다.



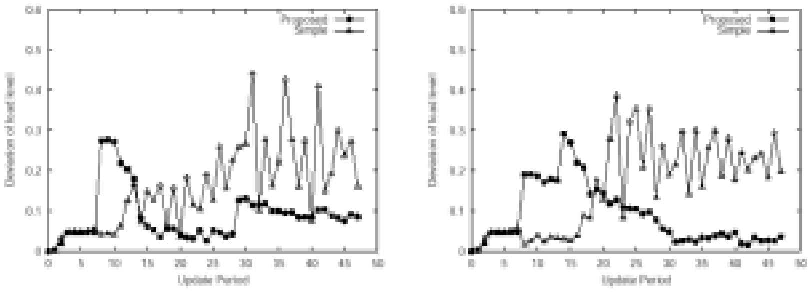
(a) Load = 6Mbps (b) Load = 5.2Mbps

(c) Load = 4.2Mbps

그림 4 CBR의 경우

4.2.2 트래픽율이 변하는 CBR 트래픽

그림 5에 트래픽율이 변하는 CBR 트래픽을 사용한 결과가 나와 있다. 그림 5 (a) 는 시간 15에서 부하가 감소하고, 30에서 부하가 다시 증가하는데, 이 때 예측을 사용하지 않은 경우는 편차가 증가하고 진동하기 시작하는 것을 알 수 있다. 예측을 사용할 경우, 시간 30에서 편차의 증가가



(a) 4.8M-3.6M-4.8M (b) 3.6M-4.8M-3.6M

그림 5 트래픽율의 변화가 있는 CBR의 경우

있지만, 진동이 없이 편차가 줄어드는 결과를 볼 수 있다. 그림 5(b)의 경우에도, 시간 15에서 부하가 증가한 후 예측을 사용하지 않은 알고리즘은 진동하기 시작하지만 예측을 사용한 경우에는 편차가 서서히 줄어드는 양상을 보이고 있다.

4.2.3 망 트래픽 기록을 이용한 트래픽

그림 7에는 NLANR 트래픽 기록을 이용하여 모의실험을 수행한 결과가 나와 있다. 사용한 기

록은 Michigan 대학과 vBNS사이의 연결에서 얻은 트래픽 기록으로서 2000년 8월 21일 12:48:13의 기록이다. 이 기록은 여러 기록중에서 트래픽 양이 많고 어느 정도 트래픽율의 변화가 있는 기록이다. 그림 6에 사용한 기록의 10초, 20초, 30초 단위의 평균과 최대값이 도시되어 있다. 여러 부하 수준에 대하여 실험하기 위하여, 모의실험 네트워크의 대역폭이 평균 트래픽율의 100%, 90%, 80%가 되도록 링크 대역폭을 수정하여 실험하였다. 실험 토폴로지와 링크 지연은 이전 실험과 동일하다. 그림 7의 가로축은 시간축으로서 한 단위가 1초이며, 이 실험에서는 30초 간격으로 링크상태 갱신이 이루어진다. 세로축은 이전 실험과 같이 링크간 트래픽의 표준편차를 나타낸다.

그림 7(a)는 망의 대역폭이 총 부하와 비슷한 경우이다. 링크 상태 갱신이 30초 간격으로 일어나기 때문에 30초 간격으로 표준편차가 크게 변하는 지점이 나타난다. 이 경우에는 망의 대역폭이 부족하여 부하의 이동이 적기 때문에, 시간이 흐름에 따라 두 알고리즘의 표준편차가 모두 일정해짐을 알 수 있다. 그림 7(b)에서는 대역폭의 여유가 있어 부하의 이동이 나타나고, 예측을 사용하지 않은 경우에는 30초 단위로 부하의 표준편차가 심하게 변함을 알 수 있다. 예측을 사용한 경우에는 CBR과 마찬가지로 표준편차의 변화가 거의 없다. 그림 7(c)의 경우는 대역폭이 더 많은 경우인데, 이 때의 결과도 그림 7(b)와 비슷하다.

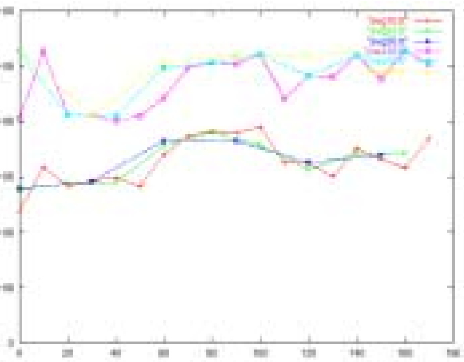
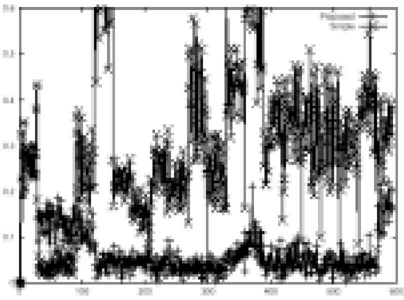
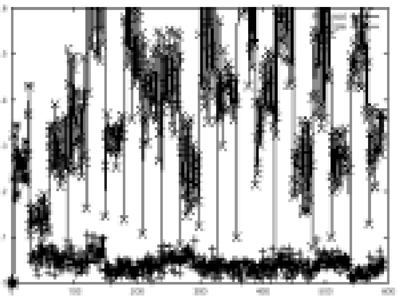


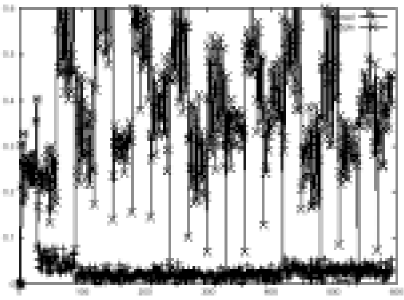
그림 6 트래픽 기록



(a) Load level = 100%



(b) Load level = 90%



(c) Load level = 80%

그림 7 트래픽 기록을 사용한 경우

5. 결론

본 논문에서는 안정적인 동적 부하 제어 기법을 제안하였다. 제안한 알고리즘은 트래픽을 다중 경로로 할당하며, 각 경로의 부하를 고려하여 망의 자원을 효율적으로 사용하였다. 예측 기법을 사용하여 현재의 망 상태와 링크 상태 메시지를 통해 전송된 상태간의 차이로 인한 진동을 줄일 수 있었다.

모의실험 결과를 통하여, 예측을 하지 않는 경우에 발생하는 불안정한 결과가 예측을 사용했을 때는 없어지는 것을 확인하였다. 또한 입력 트래픽에 변화가 있는 경우의 결과를 실험하였으며, 실제의 트래픽 트레이스를 사용한 실험에서 실제의 환경 하에서도 알고리즘이 제대로 동작하는 것을 확인하였다.

본 논문에서 연구한 것은 측정된 트래픽 매트릭스가 없는 경우의 트래픽 할당방법이다. 향후 연구 주제로는 트래픽 매트릭스를 알고 있을 경우에 동적인 부하 제어를 하는 문제가 있으며, 트래픽 엔지니어링을 위해 다중 경로를 찾는 문제가 있다.

참고문헌

[1] Don Torrieri, ''*Algorithms for Finding an Optimal Set of Short Disjoint Paths in a Communication Network,*'' IEEE Transactions on Communications, Vol. 40, No. 11,

November 1992

[2] Richard G. Ogier, Vlad Rutenburg, Nachum Shacham, ''*Distributed Algorithms for Computing Shortest Pairs of Disjoint Paths,*'' IEEE Transactions on Information Theory, Vol. 39, No. 2, March 1993

[3] Zhiruo Cao, Zheng Wang, Ellen Zegura, ''*Performance of Hashing-Based Schemes for Internet Load Balancing,*'' INFOCOM 2000

[4] J. Moy, ''*OSPF Version 2,*'' RFC2328, April 1998.

[5] R. Guerin et al., ''*QoS Routing Mechanisms and OSPF Extensions,*'' RFC 2676, August 1999.

[6] A. Shaikh, J. Rexford, and K. G. Shin, ''*Load-Sensitive Routing of Long-Lived IP Flows,*'' SIGCOMM'99, 1999

[7] UCB/LBNL/VINT Network Simulator - ns, http://www.isi.edu/nsnam/ns/

[8] ''*NLANR network traffic traces,*'' http://moat.nlanr.net/Traces/

[9] D. Thaler, C. Hopps, ''*Multipath Issues in Unicast and Multicast NextHop Selection,*'' RFC2991, November 2000

[10] E. Rosen, A. Viswanathan, R. Callon, ''*Multiprotocol Label Switching Architecture*'', work in progress, draft-ietf-mpls-arch-07.txt, July 2000.

[11] D. Thaler, and C.V. Ravishankar, ''*Using Name-Based Mappings to Increase Hit Rates,*'' IEEE/ACM Transactions on Networking, February 1998.

[12] C. Villamizar, ''*MPLS Optimized Multi-path(MPLS-OMP),*'' IETF Internet Draft(Work in Progress), Feb. 1999

[13] C. Villamizar, ''*OSPF Optimized Multi-path(OSPF-OMP),*'' IETF Internet Draft(Work in Progress), Feb. 1999