그리드 네트워크에서 종단간 전송 지연 향상을 위한 bottleneck 링크 대역폭 측정 알고리즘

최 원 석[†] · 안 성 진[††] · 정 진 욱[†††]

요 약

본 논문은 그리드 네트워크의 패킷 지연시간단축을 위한 병목 링크 대역폭 측정 알고리즘을 제안하고자 한다. 병목 링크 대역폭 측정방법으로 packet pair 알고리즘과 Paced Probes가 있으나 이들은 pacer 패킷으로 cross traffic의 영향을 줄이면서 paced probe 패킷들이 링크를 지날 때 발생하는 시간 차이를 이용하여 병목 링크 대역폭을 구한다. 그러나 경로상의 병목 링크 대역폭을 측정할 뿐이지 병목현상이 발생한 지점은 알 수 없는 문제점이 있다. 이를 hop-by-hop 측정에 기반한 병목 대역폭 측정을 제시하여 그리드 네트워크에서 전체적인 패킷 지연 개선을 이룰 수 있다. hop-by-hop의 병목 대역폭을 측정하기 위해 paced probe 패킷에 타임스탬프 옵션을 이용하며 병목현상 해소를 통하여 패킷 전송 지연의 단축을 시뮬레이션을 통해 알 수 있다. 본 논문에서 제시한 알고리즘으로 병목 링크의 대역폭과 위치를 파악하여 그리드 FTP나 실시간 QoS를 보장하는 데이터 전송에 기여할 것으로 기대된다.

Bottleneck link bandwidth Measurement Algorithm for improving end-to-end transit delay in Grid network

Wonseok Choi[†] · Seongjin Ahn[††] · Jin Wook Chung[†††]

ABSTRACT

This paper proposes a bottleneck link bandwidth measurement algorithm for reducing packet transmission delay within the grid network. There are two methods for measuring bottleneck link bandwidth : Packet Pair algorithm and Paced Probes algorithm. They measure bottleneck link bandwidth using the difference in arrival times of two paced probe packets of the same size traveling from the same source to destination. In addition, they reduce the influences of cross traffic by pacer packet. But there are some problems on these algorithms : it's not possible to know where bottleneck link occurred because they only focus on measuring the smallest link bandwidth along the path without considering bandwidth of every link on the path. So hop-by-hop based bottleneck link bandwidth measurement algorithm can be used for reducing packet transmission delay on grid network. Timestamp option was used on the paced probe packet for the link level measurement of bottleneck bandwidth. And the reducing of packet transmission delay was simulated by the solving a bottleneck link. The algorithm suggested in this paper can contribute to data transmission ensuring grid FTP and realtime QoS by detecting bandwidth and the location where bottleneck link occurred.

키워드 : Bottleneck 대역폭, NMWG, 그리드 네트워크(Grid Network, 전송 지연(Transit Delay), 타임스탬프(Timestamp)

1. 서 론

인터넷이 발전하면서 대역폭은 점차 증가하였으나 모든 사용자가 충분히 사용할 수는 없고 또한 특정 지역이나 시간에 트래픽이 폭주하여 병목 링크가 발생한다. 병목 링크는 패킷의 전송 지연을 증가시켜 그리드 네트워크의 성능을 저하 시킨다. 따라서 효과적인 그리드 네트워크 관리를 위해 bottleneck 링크의 발생 원인을 찾아 성능 개선을 하는 것이 중요한 이슈가 되었다. 네트워크 bottleneck 측정은 많은 인터넷 애플리케이션과 프로토콜, 특히 방대한 양의 데이터를 전송하는 그리드 FTP나 멀티미디어 스트림과 같은 실시간 QoS를 보장해야 하는 데이터 전송에 중요하다[5].

그리드 네트워크 성능을 측정을 위해 GGF(Global Grid Forum)의 NMWG(Nwtwork Measurement Working Group)에서는 Internet2 end-to-end initiative등의 다른 표준 기구와 협력하여 그리드 환경에서 요구되는 네트워크 특성과 측정 방법에 대한 연구를 수행하고 있다[1, 2]. 현재 대역폭 측정 기술은 경로상에 있는 각각의 링크를 측정하는 Link-level 대역폭 측정 기술[6]과 경로의 병목 대역폭을 측정하는 종단간 대역폭 측정 기술로 구분된다[7]. 이러한 대역폭을 측정하기 위한 방법들로 HTTP 처리량[8], TCP 처리량

† 준 회 원 : 성균관대학교 대학원 정보통신공학부
†† 종신회원 : 성균관대학교 컴퓨터교육과 교수
††† 종신회원 : 성균관대학교 전기전자 및 컴퓨터공학과 교수
논문접수 : 2003년 8월 8일, 심사완료 : 2003년 10월 6일

[9], 가용 대역폭[10], 병목 링크 대역폭이 있다. HTTP와 TCP가 현재 인터넷에서 많이 쓰이고 있으나 다른 애플리케이션과 전송 프로토콜은 고유의 성능 특징이 있으므로 HTTP와 TCP 처리량 방식으로는 측정할 수 없다. 가용 대역폭은 특정 애플리케이션이나 전송 프로토콜에 상관없이 측정할 수 있으나 인터넷에서 발생하는 다양한 cross traffic 때문에 정확한 측정이 어렵다. 그러나 병목 링크 대역폭은 이론적으로 잘 정의되어 있다. 예를 들면 UDP 에코 툴[16, 17]을 이용하여 패킷의 지연을 측정하여 구하는 방법[4], bprobe와 cprobe를 이용여 병목의 혼잡을 측정하는 방법[10], Packet Pair probing을 이용한 방법[11, 12] 등이 있다.

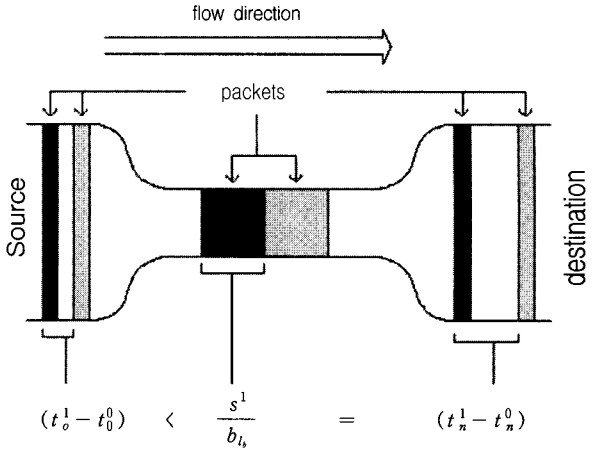
본 논문에서는 기존의 packet pair 알고리즘[4]과 Paced Probes[5]를 이용하며 probe 패킷 헤더에 타임스탬프 옵션을 사용한다. 기존의 알고리즘들은 경로상의 병목 대역폭을 측정하였으나 측정하고자 하는 링크에서 타임스탬프 값을 구하면 병목 링크의 대역폭과 함께 위치도 파악한다. 병목 링크의 위치를 파악하면 SNMP를 이용하여 병목이 발생하는 원인을 파악한 후 시스템 성능 향상이나 링크의 용량 증가를 통해 네트워크의 성능을 향상시킨다[3]. 네트워크 시뮬레이션 도구[15]를 이용하여 병목 링크가 패킷 전송 지연에 미치는 영향을 알아본다.

2. Bottleneck 대역폭 측정방법

2.1 Packet Pair Technique

Packet Pair 기술은 FIFO queuing 네트워크에서 송신지와 수신지가 같고 패킷 사이즈도 같은 두 개의 패킷이 수신지에 도착할 때 패킷간의 시간 차이를 이용하여 병목 대역폭을 구한다. Bolot, Carter와 Crovella, Paxson 그리고 Lai와 Baker는 병목 대역폭을 측정하기 위하여 Packet Pair 모델과 이론을 사용하였다. (그림 1)은 똑같은 크기의 패킷이 병목 링크를 지나갈 때 발생하는 패킷의 시간간격을 보여주고 있다. t_0^0은 0번째 노드의 0번째 패킷을 나타내고 s^1은 보내는 패킷의 크기를 나타내는데 모든 패킷의 크기는 같아야 한다. b_l은 각 링크의 대역폭을 나타낸다. Packet Pair 기술은 목적지에 도착한 패킷의 시간 차이를 이용하여 병목 대역폭을 구한다.

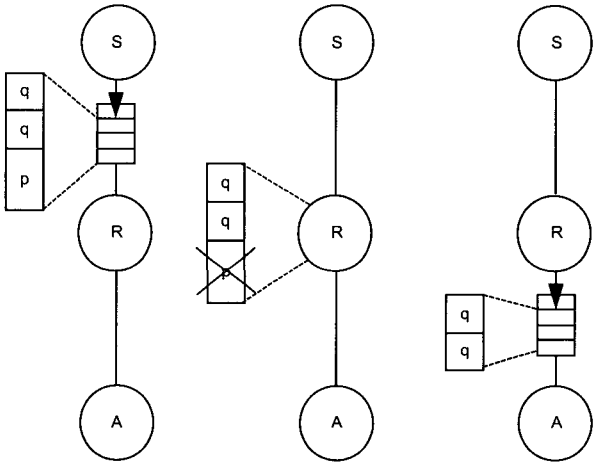
이 방정식의 근본적인 논리적 근거는 다음과 같다. 만약 두 개의 패킷이 충분히 가까운 시간 간격을 두고 보내졌을 때 병목 링크 구간이 없는 경로에서는 두 패킷의 목적지 도착 시간 간격은 보낼 때 시간 간격과 같을 것이다. 그러나 경로 중에 병목 링크 구간이 존재한다면 패킷들은 병목 링크 구간의 큐에 쌓이므로 $\left(\frac{s_1}{b_l} > t_0^1 - t_0^0\right)$ 처음 보낼 때의 시간 간격에서 벌어지므로 목적지에 다른 시간 간격을 두고 도착할 것이다 $(t_n^1 - t_n^0)$[14].



(그림 1) 병목 대역폭 측정

2.2 Paced Probes

Packet pair 알고리즘은 cross traffic의 영향을 안 받는다고 가정한다. 그러나 실제 네트워크에서는 여러 호스트에서 발생하는 다양한 cross traffic이 발생하여 정확한 측정값을 구할 수 없다. Cross traffic의 영향을 줄이기 위해 패킷 사이즈가 큰 pacer 패킷을 사용한다. Pacer 패킷의 역할은 전송 중에 발생하는 packet pair의 지연 간격을 없애는데 있다. 크기가 큰 Pacer 패킷이 각 노드의 큐에 저장되면서 packet pair 사이의 간격을 없애면서 측정 경로의 시작점까지 전송된다. (그림 2)와 같이 Pacer packet이 노드 R에서 제거될 때 packet pair는 back-to-back으로 남아있고 그때 남아있는 경로의 병목 대역폭을 구할 수 있다.



(그림 2) Pacer packet

Pacer 패킷을 사용하면 네트워크 내 하위 경로의 병목 대역폭을 구할 수는 있지만 목적지는 항상 A 지점이어야 한다. 만약 목적지가 A 지점이 아닌 다른 지점까지의 병목

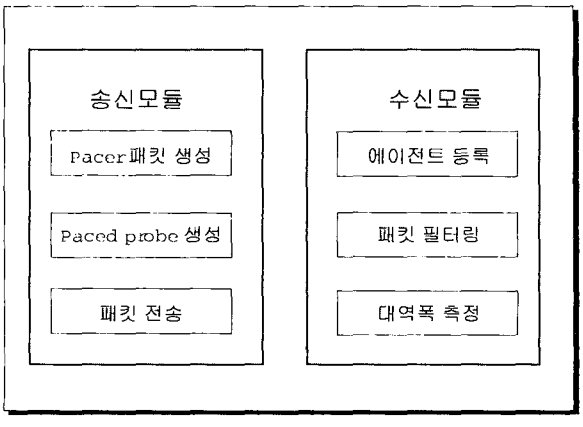
대역폭을 구하려 할 때, 에이전트 시스템은 각 노드에 존재해야 한다. 이는 네트워크의 전체 성능을 떨어뜨리고 많은 비용이 소모될 뿐만 아니라 실제 네트워크 환경에서는 불가능하다. 이러한 문제점을 해결하기 위해 본 논문에서는 paced probe 패킷에 IP 헤더 옵션 중 타임스탬프를 이용한다. 측정하고자 하는 링크의 시작 노드와 끝나는 노드의 타임스탬프 값을 이용하여 packet pair 사이에 발생한 시간 간격을 알 수 있으므로 해당 링크의 병목 대역폭을 측정할 수 있다.

3. 병목 링크 대역폭 및 위치 측정

3.1 패킷 생성 에이전트

패킷 생성 에이전트는 네트워크의 각 종단 시스템에 존재하여 pacer 패킷과 paced probe 패킷을 생성한다. Pacer 패킷은 사이즈가 큰 패킷으로 목적지는 측정하고자 하는 링크의 시작 노드이다. Pacer 패킷은 지나가는 경로마다 버퍼에 쌓여 paced probe에 cross traffic이 삽입되어 병목 링크의 잘못된 측정을 예방한다. Paced probe는 병목 링크 대역폭을 측정하기 위한 패킷으로 cross traffic의 영향을 받지 않고 측정하고자 하는 링크에서 back to back으로 있어야 정확한 측정을 할 수 있다. 에이전트 시스템은 정확한 크기와 목적지를 갖는 패킷을 생성하고 보내야 한다.

수신측 에이전트는 네트워크 모니터링을 통해 패킷을 캡쳐하는 기능이다. 이때 네트워크에 돌아다니는 모든 패킷을 캡쳐하기에는 너무 많은 자원낭비가 발생된다. 따라서 각 에이전트를 등록하여 등록된 에이전트에서 보낸 패킷만을 받는다. 패킷을 받으면 IP 헤더의 타임스탬프 값을 검사하여 병목 링크 대역폭 및 위치를 측정한다. 에이전트 시스템은 송신측과 수신측 에이전트로 구분하지 않고 하나의 에이전트 시스템에서 패킷 생성과 캡쳐를 수행한다. (그림 3)은 에이전트의 구조를 보여주고 있다.



(그림 3) 에이전트 시스템

3.2 병목 링크 대역폭 및 위치 측정 알고리즘

기존의 병목 대역폭 측정 방법은 경로 상에 존재하는 모든 링크 중에서 가장 낮은 대역폭을 갖는 링크의 대역폭을 구하여 병목 대역폭을 측정하였다. 네트워크에 병목 현상이 발생했을 경우 단순한 대역폭 측정으로는 병목 현상을 해결할 수 없다. 정확한 병목 링크의 위치를 측정하여 병목 원인을 해소해야 패킷 지연이 감소한다. 이를 위해 기존의 paced 패킷에 타임스탬프 옵션을 사용하여 모든 경로 상에 존재하는 링크의 대역폭을 구한다. paced 패킷은 네트워크의 노드를 지날 때 마다 시간을 기록하고 송신측에서 노드와 노드를 지날 때 발생하는 패킷의 시간 차이를 이용하여 링크의 대역폭을 구한다. 이러한 방식으로 end-to-end 간 병목 대역폭을 측정하면 대역폭 뿐만 아니라 정확한 위치도 측정이 가능하여 병목 원인을 해소할 수 있어 패킷의 전송 지연도 줄일 수 있다. (그림 4)는 네트워크에서 링크별 병목 대역폭을 구하는 알고리즘이다.

```
while (k = 0 < M)
{
  for (j = 1; j ≤ N; j++)
  {
    for (i = 2; i ≤ L; i++)
    {
      create : p^1_{R_i,R_{i-1}}, P^2_{R_i,R_{i-1}}, q;
        s(p^1_{R_i,R_{i-1}}) = s(p^2_{R_i,R_{i-1}}) = <<s(q);
      send : D(p^1_{R_i,R_{i-1}}) = D(p^2_{R_i,R_{i-1}}) = R_n, D(q) = R_{i-1};
      receive : p^1_{R_i,R_{i-1}}, p^2_{R_i,R_{i-1}};
        I^{p^1_{R_i,R_{i-1}} p^2_{R_i,R_{i-1}}}_{R_i} = (M^{p^1_{R_i,R_{i-1}}}_{R_i} - M^{p^1_{R_i,R_{i-1}}}_{R_{i-1}}) - (M^{p^2_{R_i,R_{i-1}}}_{R_i} - M^{p^2_{R_i,R_{i-1}}}_{R_{i-1}});
        if(I^{p^1_{R_i,R_{i-1}} p^2_{R_i,R_{i-1}}}_{R_i} ≤ 0) B_i = B_{i-1};
        B_i = s(p^1_{R_i,R_{i-1}}) / I^{p^1_{R_i,R_{i-1}} p^2_{R_i,R_{i-1}}}_{R_i};
    }
  }
  k++;
}
```

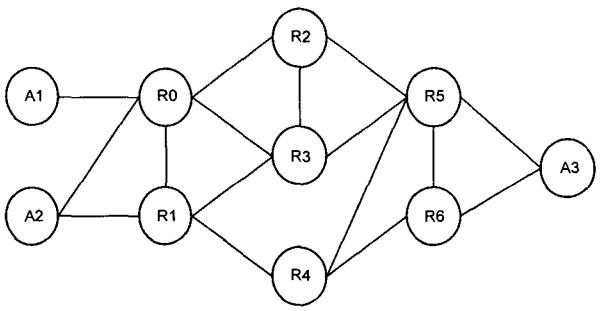
(그림 5) 병목 링크 대역폭 및 위치 측정 알고리즘

변수 M은 패킷을 보내는 방향의 에이전트 개수이며 N은 받는 쪽 에이전트 개수를 의미한다. 송신 에이전트마다 모든 수신 에이전트에 패킷을 전송하여 병목 링크 대역폭을 구한다. 예를 들면 L_i번째 링크의 병목 대역폭을 구하기 위하여 송신지 에이전트는 pacer 패킷 q와 paced 패킷 $p^1_{R_i,R_{i-1}}$, $p^2_{R_i,R_{i-1}}$을 생성한다. pacer 패킷의 목적지는 R_{i-1}이고 paced probe 패킷은 R_i이다. 패킷 $p^1_{R_i,R_{i-1}}$와 $p^2_{R_i,R_{i-1}}$은 생성 시에 IP 헤더의 타임스탬프 옵션 필드를 이용하기 위하여 R_{i-1}과 R_i 라우터의 IP 주소를 입력하여 그 지점에서의 패킷 도착 시간을 기록한다. Pacer 패킷을 사용함으로써 paced 패킷은

cross traffic의 영향을 덜 받으면서 R_{i-1} 지점까지 도착한다. R_{i-1}지점에서 pacer 패킷은 제거되고 두 개의 paced 패킷이 타임스탬프 옵션 필드에 현재 라우터의 시간을 기록하면서 수신 에이전트에 도착한다. 수신 에이전트는 타임스탬프 옵션 필드에 기록된 시간을 이용하여 링크 L_i 상에서 두 패킷의 시간 차이를 나타내는 $I^{p^1_{R_{i-1}R_i}, p^2_{R_{i-1}R_i}}_{R_{i-1}R_i}$ 값을 구한다. 만약 $I^{p^1_{R_{i-1}R_i}, p^2_{R_{i-1}R_i}}_{R_{i-1}R_i}$ 값이 0 보다 작거나 같으면 두 패킷의 시간 간격이 발생하지 않은 것이므로 L_i의 병목 대역폭은 전 링크의 병목 대역폭과 같다고 한다. 반면에 $I^{p^1_{R_{i-1}R_i}, p^2_{R_{i-1}R_i}}_{R_{i-1}R_i}$ 값이 0보다 크다는 의미는 두 패킷 사이에 시간 간격이 벌어졌다는 의미이고 이때의 병목 대역폭 $B_i = \frac{s(p^1_{R_{i-1}R_i})}{I^{p^1_{R_{i-1}R_i}, p^2_{R_{i-1}R_i}}_{R_{i-1}R_i}}$이다.

4. 시뮬레이션 및 결과

이번 실험의 목적은 병목 링크의 원인을 찾아 해결하였을 때 패킷 전송 지연의 차이를 알아보려 한다. 병목현상이 발생하는 원인을 크게 두 가지로 나눌 수 있는데 하나는 링크의 용량이 부족이고 다른 하나는 호스트의 저성능이다. 병목 링크를 발견하면 SNMP를 이용하여 회선의 이용률을 구한다. 회선의 이용률을 구하는 계산식은 (max(ifInOctets, ifOutOctets)*8*100)/((sysUpTime/100)*LineSpeed)이다[18]. 회선의 이용률이 높은데도 병목이 생기면 링크 대역폭이 작은 것이며 이용률이 낮으면 호스트의 성능이 낮아서 들어오는 패킷 처리율이 낮아 병목이 발생한다. 실험을 위해 네트워크 시뮬레이션 도구[15]를 사용하여 (그림 5)와 같은 네트워크를 구축한다. 송신측 A1과 A2에서 수신측 A3 사이에 존재하는 모든 링크의 대역폭을 측정하여 병목을 발생하는 링크를 찾아 병목 원인을 해결하여 패킷의 전송 지연을 줄인다. 링크의 대역폭 측정을 위해 IP 타임스탬프 옵션을 사용하여 각 노드에서의 도착 시간을 구한 후 제안된 알고리즘으로 링크의 대역폭을 측정한다. 출발링크의 cross traffic은 aggregated Pareto ON/OFF UDP이며 burst의 평균 시간은 10msec, idle 평균 시간은 80msec이다. 그리고



(그림 5) 네트워크 구성도

발생된 패킷의 평균 사이즈는 512바이트이고 최대 크기는 1500바이트이다. 에이전트에서 사용한 pacer 패킷 크기는 1500바이트이며 Paced probe 패킷 크기는 1000바이트이다. Cross traffic에 의해 영향을 줄이기 위해 <표 1>에서 나온 측정값은 패킷 전송을 10000번 수행하여 얻은 값의 평균값이다.

① 시나리오 1 : 링크 전송률은 200Kbps이고 링크의 대역폭은 100Mbps이며 병목 대역폭은 10Mbps이다. <표 1>에서 2번째 열은 크기가 8Mbyte 패킷을 전송할 때 걸리는 시간을 나타내고 3번째 열은 각 링크의 종단점을 4, 5번째 열은 A1, A2에서 A3까지 모든 링크의 실제 대역폭과 측정된 대역폭을 보여주고 있으며 단위는 Mbps이다.

〈표 1〉 낮은 링크 대역폭 결과

경 로	시 간	링 크	링크 성능	측정값
(A1, A3)	3.63	(A1, R0)	100	78
		(R0, R3)	100	78
		(R3, R5)	10	9.73
		(R5, A3)	100	77
(A2, A3)	2.83	(A2, R1)	100	78
		(R1, R4)	10	9.75
		(R4, R6)	100	79
		(R6, A3)	100	78

우리는 <표 1>을 통하여 (R3, R5)와 (R1, R4)가 병목 링크라는걸 알 수 있다. 병목 링크의 이용률은 98%와 94%이다. 따라서 링크의 용량을 늘려주면 병목이 해결된다. (R3, R5)와 (R1, R4)의 대역폭을 100Mbps로 하고 같은 크기의 패킷을 전송한다. 실험결과 (A1, A3)는 1.58sec 빠른 2.05sec이고 (A2, A3)는 0.79sec 빠른 2.04sec이다.

② 시나리오 2 : 호스트 R3, R4의 링크 전송률은 20Kbps이고 모든 링크의 대역폭은 100Mbps이다.

〈표 2〉 낮은 호스트 성능

경 로	시 간	링 크	링크 성능	측정값
(A1, A3)	4.32	(A1, R0)	100	78
		(R0, R3)	100	78
		(R3, R5)	100	7.89
		(R5, A3)	100	77
(A2, A3)	4.19	(A2, R1)	100	78
		(R1, R4)	100	76
		(R4, R6)	100	7.13
		(R6, A3)	100	78

병목는 (R3, R5)와 (R4, R6)이며 이용률은 28%와 31%이

다. 따라서 호스트의 성능이 낮아서 병목이 발생했으며 해당 호스트의 링크 전송률을 200Kbps로 높여주면 네트워크의 성능이 향상된다. 실험결과 (R3, R5)에서는 2.01sec 빠른 2.31sec이고 (R4, R6)은 1.87sec 빠른 2.32sec이다.

5. 결 론

본 논문은 병목 링크 대역폭 측정을 통해 병목이 패킷 전송 지연에 미치는 영향을 분석해 네트워크의 성능을 향상시키고자 한다. 시뮬레이션을 통해 본 논문에서 제안한 알고리즘으로 각 경로의 링크 대역폭을 구하여 전송경로의 병목 링크 대역폭과 병목 지점을 파악한다. 병목 지점을 파악한 후 SNMP를 이용하여 링크의 이용률을 구해 병목의 원인을 파악하고 해결한다. 실험결과 시스템의 성능 저하나 링크의 용량이 작아 병목 링크가 발생할 때 패킷 전송 시간보다 병목 원인을 해결하고 난 후의 패킷 전송 지연이 더 향상된 것을 알 수 있다. 따라서 병목 링크가 패킷 전송 지연에 영향을 주고 병목 원인 해결이 네트워크의 성능을 향상시킨다. 본 논문에서 병목 링크의 대역폭과 위치를 구하기 위해 동적 병목 링크 측정 알고리즘을 사용하였다. 기존의 packet pair 알고리즘은 크기가 같은 두 패킷이 전송 중에 발생한 시간 간격을 이용해 대역폭을 구하고, pacer packet은 cross traffic의 영향을 덜 받게 한다. 그러나 경로의 병목 링크 대역폭을 측정할 뿐 병목 지점을 알 수 없다. 그러나 probe 패킷에 타임스탬프 옵션을 사용하면 패킷이 지나가는 모든 링크의 대역폭을 구할 수 있어 병목 링크의 대역폭뿐만 아니라 병목 지점도 파악할 수 있다. 병목 지점을 파악하면 SNMP를 이용하여 병목의 원인을 해결하여 패킷 전송 지연을 줄일 수 있어 네트워크의 전체 성능을 향상시킬 수 있다. 본 논문에서 제안한 알고리즘과 SNMP를 이용하면 NMWG에서 제안한 그리드 네트워크 대역폭 성능을 측정할 수 있다. 또한 그리드 FTP에서 대용량의 파일을 다운받을 때 병목 링크 대역폭 측정은 가장 빠른 패킷 전송 지연을 제공하는 서버를 선택할 수 있게 하며 QoS를 제공하기 위해서도 필요한 정보이다.

참 고 문 헌

[1] http://www.gridforum.org/.

[2] http://www-didc.lbl.gov/NMWG/.

[3] 신상철, 안성진, 정진욱, "SNMP를 이용한 인터넷 분석 파라미터 추출 시스템의 설계 및 구현", 정보처리학회지, 제6권 제3호, pp.710-721, 1999.

[4] Jean-Chrysostome Bolot. "End-to-end packet delay and loss behavior in the internet," In *Proceeding of the 1996 USENIX Technical Conference*, January, 1996.

[5] Khaled Harfoush and Azer Bestavros and John Byers, "Measuring Bottleneck Bandwidth of Targeted Path Segments," TECHNICAL REPORT BUCS-2001-016, Boston University, July, 2001.

[6] A.Barbir et al. Known CDN Request-Routing Mechanisms. http://www.globecom.net/ietf/draft/draft-cain-cdnp-known-request-routing-01.html, February, 2001.

[7] D. Andersen, H. Balakrishnan, M. F. Kaashoek and R. Morris, "Resilient Overlay Networks," In *Proceedings of SOSP 2001(to appear)*, October, 2001.

[8] Mark R. Stemm, "An Network Measurement Architecture for Adaptive Applications," PhD thesis, University of California, Berkeley, 1999.

[9] M. Mathis and J. Mahdavi, "Diagnosing Internet Congestion with a Transport Layer Performance Tool," In *Proceedings of INET*, 1996.

[10] Robert L. Carter and Mark E., "Crovella. Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks," Technical Report BU-CS-960007, Boston University, 1996.

[11] Srinivasan Keshav, "A Control-Theoretic Approach to Flow Control," In *Proceedings of ACM SIGCOMM*, 1991.

[12] Vern Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," PhD thesis, University of California, Berkeley, April, 1997.

[13] Kevin Lai and Mary Baker, "Measuring Bandwidth," In Proceedings of IEEE INFOCOM, March, 1999.

[14] Kevin Lai and Mary Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay," In *Proceedings of ACM SIGCOMM*, August, 2000.

[15] ns : Network Simulator, http://www-mash.cs.berkeley.edu/ns/ns.html.

[16] D. Sanghi, A. K. Agrawala, B. Jain, "Experimental assessment of end-to-end behavior on Internet" Proc. IEEE Infocom '93, San Fransisco, CA, pp.867-874, March, 1993.

[17] D. sanghi, O. Gudmundsson, A. K. Agrawala, "Study of network dynamics Proc," 4th Joint European Net-working Conference, Trondheim, Norway, pp.241-249, May, 1993.

[18] 안성진, 정진욱, "SNMP MIB-II를 이용한 인터네트 분석 파라미터 계산 알고리즘에 관한 연구", 정보처리학회지, 제5권 제8호, pp.2102-2116, 1998.

최 원 석

e-mail : wschoi@songgang.skku.ac.kr

2002년 성균관대학교 정보공학과(학사)

2002년~현재 성균관대학교 전기전자 및 컴퓨터공학부 대학원 석사 과정

관심분야 : 네트워크 관리, 트래픽 분석, 임베디드 리눅스

안 성 진

e-mail : sjahn@comedu.skku.ac.kr

1988년 성균관대학교 정보공학과(학사)

1990년 성균관대학교 대학원 정보공학과 (석사)

1990년~1995년 한국전자통신연구원 연구원 전산망 개발실 연구원

1996년 정보통신 기술사 자격 취득

1998년 성균관대학교 대학원 정보공학과(박사)

1999년~현재 성균관대학교 컴퓨터교육과 조교수

관심분야 : 네트워크 관리, 트래픽 분석, Unix 네트워킹

정 진 욱

e-mail : jwchung@songgang.skku.ac.kr

1974년 성균관대학교 전기공학과 학사

1979년 성균관대학교 대학원 전자공학과 석사

1991년 서울대학교 대학원 계산통계학과 박사

1982년~1985년 한국과학기술 연구소 실장

1981년~1982년 Racal Milgo Co. 객원연구원

1985년~현재 성균관대학교 전기전자 및 컴퓨터공학부 교수

관심분야 : 컴퓨터 네트워크, 네트워크 관리, 네트워크 보안