2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

Web 기반 bottleneck 구간 탐지 및 분석 시스템의 설계

성연국*, 조강홍*, 최영수*, 정진욱*
*성균관대학교 전기전자 및 컴퓨터 공학부
e-mail:qbaby@kebi.com

Design of Bottleneck detection and analysis System on Web-base

Yun-Gook Sung*, Kang-Hong Cho*, Young-Su Choi*, Jin-Wook Chung*
*School of Electrical and Computer Engineering, Sungkyunkwan University

요약

본 논문은 Client & Server 모델을 기반으로 선로 이용률과 응답시간을 분석, 조합하여 웹 상에서 해당 목적지 경로 사이에 있을 수 있는 bottleneck 구간을 탐지하기 위한 시스템을 설계하였다. 이를 위해 traceroute를 이용하여 사용자와 해당 목적지 경로 사이의 인터페이스들을 추적하였다. 추적된 인터페이스들의 IP로 TCP/IP 표준 네트워크 망 관리 프로토콜인 SNMP polling을 하여 인터페이스로부터 각종 MIB 정보들을 가져와 각 인터페이스간의 선로 이용률을 분석해냈으며, 또한 정확한 bottleneck 구간을 알아내고자 각 라우터로 ping을 실행시켜 응답시간을 계산하였다. 이렇게 얻은 선로 이용률과 응답시간으로 보다 정확한 트래픽 정보와 bottleneck 구간을 분석해낼 수 있었다. 이러한 분석 시스템으로 인해 사용자들은 전문적인 지식 없이 단지 자신의 컴퓨터의 웹브라우저를 통한 서버 분석 시스템 접속만으로도 해당 구간의 어떤 부분이 bottleneck 구간인지를 탐지, 분석할 수 있을 것이다.

1. 서론

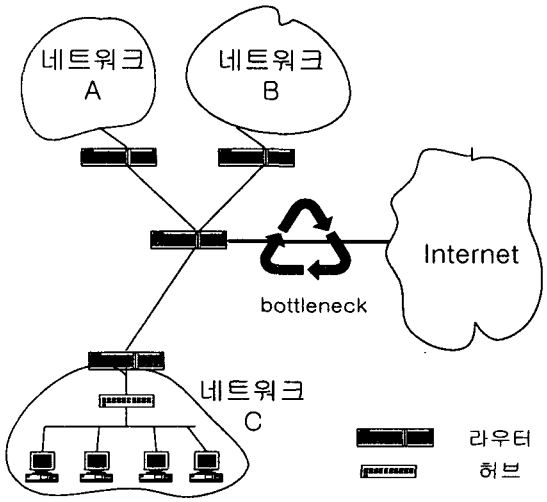
급격한 정보화로 인해 컴퓨터 네트워크의 기술은 요구와 필요에 의해 나날이 발전을 거듭해왔다. 특히, TCP/IP를 기반으로 하는 인터넷은 우리 생활의 일부분으로 자리잡게 되었다. 우리는 인터넷을 통해 WWW(world wide web), e-mail, 화상채팅, 등의 각종 서비스를 제공받는다. 이중 예술, 컴퓨터, 교육, 뉴스, 날씨, 관광정보, 또는 각종 전문적인 정보들을 탐색하고 얻을 수 있는 WWW는 인터넷을 사용하는 대부분의 사람들이 생활의 일부로서 활용하고 있다

이러한 네트워크 사용자들의 급속한 증가와 점차 커지는 응용 서비스들이 발생시키는 트래픽은 네트워크 상의 성능 저하 발생 확률을 크게 급증시키고 있다. 결과적으로 네트워크 망에서의 정보 지연 원인이 되는 bottleneck 구간 문제가 발생하고 있다.

대개 bottleneck 현상은 사용자의 PC 나 LAN상에서의 문제도 있을 수 있겠으나 대부분의 경우는 네트워크 망에서의 과도한 트래픽으로 인해 일어난다. 그러나 매우 방대하고 복잡한 네트워크망의 어느 지점에서 bottleneck 현상이 발생하는지, 또 어느정도의 신뢰성을 가지는 지는 PC사용자나 네트워크 관리자로서는 현재로서 정확히 탐지하기가 쉽지 않다. 단지 ping이나 traceroute를 실행해 응답시간을 측정, 분석함으로써 어느 특정한 단일 구간의 탐지만이 가능한 상태이다. 이러한 상황으로부터 전체적인 네트워크 망에서의 선로 이용률과 응답시간 기반의 bottleneck 구간 탐지 및 분석을 가능하게 하고자 웹 기반 네트워크 망에서의 bottleneck 구간 분석 시스템이 필요하게 되었다.

본 논문에서는 네트워크 망에서의 bottleneck구간 탐지 알고리즘을 제시하였고, 이글을 기반으로 web기반 bottleneck 구간 탐지 및 분석 시스템을 설계하였다.

2. 네트워크 상의 bottleneck 현상



[그림 1] bottleneck 발생의 예

네트워크 상의 bottleneck이란 주로 이용량의 급증이나 선로가 통합될 때 트래픽의 선로 한계량 초과로 인한 정보 지연 현상을 말한다. 그 예를 [그림 1]에서 보여주고 있다. 그 외 경로 상 라우터의 낮은 CPU 처리속도 문제나 buffer 한계량 초과 문제에서도 bottleneck은 야기될 수 있다. 이와 같은 bottleneck은 네트워크 상의 성능 저하를 일으키며 네트워크의 QoS(Quality of Service)를 떨어뜨리는 중요한 요소로 작용한다.

3. bottleneck 구간 탐지 알고리즘

bottleneck 구간을 탐지하기 위한 접근 방법으로 선로 이용률과 각 라우터에 대한 패킷 응답시간 이 두 요소를 분석, 조합하여 bottleneck 구간을 탐지할 수 있다. 단일 라우터에 대한 응답시간 뿐만이 아니라 MIB의 정보를 이용한 선로 이용률 계산은, 간단하며 쉽게 데이터를 얻을 수 있기 때문에 [표 1]에서 제시한 bottleneck 구간 탐지 알고리즘을 사용했다.

단계 i] 여기서 traceroute를 실행했을 때, 각 라우터로부터 되돌아온 ICMP 메시지의 발신지 IP 주소는 라우터의 UDP 데이터그램이 도착한 인터페이스의 IP 주소임을 정의한다. 일반적으로 라우터는 그곳의 인터페이스 IP주소를 리턴한다. 식(1)에서 처럼 경로 전체의 IP를 추적한다.

단계 ii] ping 프로그램은 ICMP 에코 요구 메시지를 해당 호스트로 보내고, ICMP 에코 응답이 돌아오기를 기대하는데 이때 에코 응답이 도착한 시간과 에코 요구 메시지를 보낸 시간의 차를 응답시간이라 한다.[1]. 식(2)를 이용해 응답시간을 계산한다. 그런데 응답시간을 단 한 시점만을 측정해서는 정확한 트래픽을 알기 어려우므로, 식(3)을 이용해서 여러번 측정해서 계산한 그 평균값을 취한다. 식(3)의 $R_{avg}(i)$는 인터페이스의 시점 i와 i+1 사이의 평균 Response time이다.

[표 1] bottleneck 구간 탐지 알고리즘

【bottleneck 구간탐지 알고리즘】

단계 i] traceroute를 이용해 사용자와 해당 목적지 경로 사이의 인터페이스 IP를 추적한다.

$IP_{total} = \{IP_1, IP_2, IP_3, IP_4, \cdots, IP_n\}$ 식(1)

(n : 경로상의 홉 카운트)

단계 ii] 단계 i 에서 구한 각 인터페이스의 IP에 대한 응답시간을 ping을 써써 분석한다.

$R_t(i) = Ping\{IP_{total}\} \quad (1 \le i \le n)$

$= Ping\{IP_1, IP_2, IP_3, IP_4, \cdots, IP_n\}$

$= ICMP\{\text{echo reply time} - \text{echo request time}\}i$ 식(2)

$$R_{avg}(i) = \frac{\sum_{k=0}^{m} Rt(\frac{kc}{m} + i)}{m}$$ 식(3)

(m : 일정 시간 사이의 ping 카운트)

(c : 일정 시간 사이의 ping 실행 간격)

단계 iii] 역시 단계 i 에서 구한 각 인터페이스의 IP를 이용해 네트워크 구간의 선로 이용률을 분석한다.

$$U_t(i)(FDX) = \frac{\Delta MAX(\text{if } InOctets[i+1,i], \text{if } OutOctets[i+1,i]) * 8}{\Delta sysUpTime[i+1,i] * 100 * \text{if } Speed}$$

$$U_t(i)(HDX) = \frac{(\Delta \text{if } InOctets[i+1,i] + \Delta \text{if } OutOctets[i+1,i]) * 8}{\Delta sysUpTime[i+1,i] * 100 * \text{if } Speed}$$ 식(4)

단계 iv] 단계 ii 와 iii 에서 구한 Rt(응답시간)과 Uti(선로 이용률)을 비교함으로써 네트워크 경로상의 상대적인 bottleneck 구간을 추정할 수 있다.

단계 iii] 역시 단계 i 에서 구한 각 라우터의 IP를 이용해 네트워크 구간의 선로 이용률을 구한다.

① 필요한 MIB 정보들의 특성

선로 이용률을 계산하기 위해서는 각 인터페이스 그룹의 ifInOctet, ifOutOctet, ifType, sysUpTime, ifSpeed 등이 필요하며 각 변수들에 대한 설명은 [표 2]과 같다.[2].

[표 2] 몇가지 MIB 변수들의 이름과 설명

변수이름	설명
ifInOctets	프레임의 구성 문자를 포함하여 인터페이스에 수신된 옥테트의 총 수를 나타내는 카운터
ifOutOctets	프레임의 구성 문자를 포함하여 인터페이스를 벗어나서 전송되는 옥테트의 총 수를 나타내는 카운터
ifType	프로토콜 스택에 있는 네트워크 계층 바로 아래의 물리/링크 계층 프로토콜에 따라 구별되는 인터페이스의 유형을 나타내는 정수(integer)
sysUpTime	시스템의 네트워크 관리 부분이 마지막으로 재 초기화된 이후의 시간(1/100초 단위)을 나타내는 Timeticks
ifSpeed	BPS로 인터페이스의 현재 대역폭을 나타내는 게이지(Gauge)

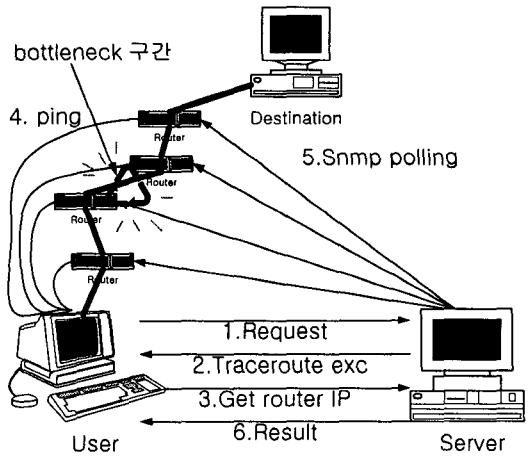
② 선로 이용률 계산

SNMP polling을 해 얻은 다섯 개의 변수와 식(4)를 이용, 계산해서 선로이용률을 구할수 있다. 선로의 대역폭을 기준으로 선로의 이용량을 백분율로 나타내는데 선로의 형식에 따라서 두가지 방식으로 구한다. 우선 MIB의 ifType의 데이터를 보고 두 방식을 구분할 수 있다. 전이중 방식(FDX)의 선로는 동시에 송수신이 일어나므로 송신 바이트량과 수신 바이트량 중 최대값으로 이용률을 계산하며(식 (4)의 FDX 이용), 단향 통신이나 반이중 방식의 선로는 한순간에 한 방향으로만 전송이 가능하기 때문에 송신 바이트량과 수신 바이트량을 더해서 이용률을 계산한다. 이와같은 계산으로 하루를 기준으로 한 시간대별 임의 지점 i로부터 지점 (i+1)로의 선로 이용률을 계산할 수 있다.[3].

단계 iv] 각 경로 라우터 간의 응답시간 및 선로 이용률을 그래프로 표시해 사용자가 쉽게 구간별 트래픽 정도를 분석할 수 있다. 이렇게 분석한 경로중 다른 경로보다 다소 느리다고 판단된 구간을 bottleneck 구간이라 판별할 수 있다.

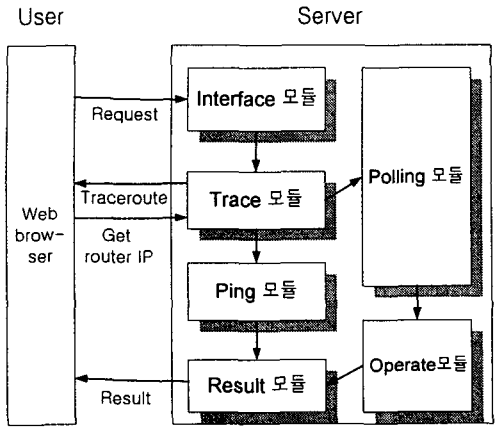
3. 탐지 시스템의 설계

시스템의 사용자 기준을 시스템의 관리자로 할 수도 있겠지만, 시스템의 보다 많은 효율을 위해 기초적인 컴퓨터 지식만을 가진 일반 개인을 대상으로 하기 위해 웹기반의 시스템을 구축하게 되었다. 기본적으로 사용자는 넷스케이프나 인터넷 익스플로어러와 같은 웹 브라우저를 갖고, 전문적인 지식없이 단지 서버 시스템의 접속만을 통해 서비스를 제공 받을 수 있다. 그런데 이때 자바 애플릿의 보안상의 문제로 사용자 PC의 사용 권한은 서버쪽이 갖는 것으로 간주한다. 왜냐하면 ping과 traceroute를 서버쪽의 프로그램이 자바 애플릿으로 실행해야 되는데, 익스플로어러나 넷스케이프 같은 웹브라우저는 기본적으로 그러한 애플릿의 접근을 허용하지 않기 때문이다. 따라서 사용자 PC의 웹브라우저 사용 권한을 서버측에서 관리함으로써 그러한 보안문제를 해결했다. 즉 사용자 웹브라우저의 보안설정 시 인증되지 않은 자바 애플릿을 실행할 수 있도록 설정했다.



[그림 2] 탐지 분석시스템의 전체 구성도

[그림 2]에서 보는 것 같이 사용자가 자신의 PC를 이용해 서버의 시스템에 접속을 하면 서버의 시스템은 bottleneck 구간 여부를 알기 원하는 해당 목적지를 사용자로부터 입력받게 된다. 이렇게 입력받은 목적지의 주소를 갖고 서버의 시스템은 traceroute를 이용해 사용자로부터 해당 목적지까지의 라우터 경로를 추적하게 된다. 이렇게 얻은 라우터들의 IP를 갖고 각 라우터로 SNMP polling을 해 얻은 각종 MIB 정보를 조합해 각 라우터들 사이의 선로 이용률을 알 수 있다. 또 사용자 PC로부터 각 라우터로 ping을 써서 패킷 응답시간도 알 수 있다. 이렇게 얻은 선로 이용률과 패킷 응답시간은 bottleneck 구간을 결정 짓는 중



[그림 3] 서버의 기능 구성도

요한 요소이다. 이것들을 분석해 해당 경로상의 bottleneck 구간을 판별할 수 있다.

이와 같은 역할을 수행하기 위해 서버는 [그림 3]과 같은 모듈의 집합으로 구성되어 있다. 각 모듈들의 기능은 다음과 같다.

① Client 모듈 : 웹 기반의 사용자 인터페이스를 포함하므로 애플릿으로 구현되었으며 서버와 서로 통신한다.

☐ interface 모듈 : 사용자와 서버 시스템을 연결시켜주는 인터페이스 기능을 수행하며 사용자가 원하는 해당 목적지의 IP또는 DNS 네임을 trace 모듈로 전달해 준다.

☐ trace 모듈 : 원격지의 사용자 컴퓨터의 traceroute application을 실행시켜 사용자와 사용자가 원하는 해당 목적지 사이의 라우터들의 IP를 구한다. 이렇게 구한 라우터들의 IP를 ping 모듈과 polling 모듈로 전달한다.

☐ ping 모듈 : 원격지의 사용자 컴퓨터의 ping application을 실행시켜, 구해진 각 라우터에 대한 응답시간을 측정한다. 구해진 응답시간을 result 모듈로 전달한다.

☐ result 모듈 : operate 모듈로부터 분석된 응답시간과 선로 이용률에 대한 정보를 전달받아 그림과 텍스트 형식으로 결과를 사용자에게 출력한다.

② Server 모듈 : application으로 구현되었으며 클라이언트와 서로 통신한다.

☐ polling 모듈 : 구해진 라우터들의 IP를 이용해 각 라우터로 snmp_polling을 해 선로 이용률을 구하기 위한 MIB정보들를 가져온다.

☐ operate 모듈 : MIB정보들을 전달받아 선로 이용률을 구한다. 구해진 선로 이용률을 result 모듈로 전달한다.

4. 결론

본 논문은 TCP/IP 표준 네트워크망 관리 프로토콜인 SNMP, ping, traceroute를 이용하여, Client & Server 모델을 기반으로 네트워크 상에서의 정보 지연의 원인을 규명하고 bottleneck 구간의 탐지 여부를 사용자에게 보여주기 위해 설계되고 구현되었다. 사용자는 SNMP나 traceroute와 같은 기타 전문 지식의 필요없이 단지 자신의 PC의 웹브라우저만을 통해 서버 시스템에 접속하므로써 자신이 접속하려는 사이트의 정보지연 문제점과 bottleneck 구간을 쉽게 판별할 수 있을 것이다.

지금과 같은 고속의 정보화 시대에 네트워크의 사용자는 급증하고 네트워크의 규모 또한 복잡해지고 방대해짐에 따라 네트워크상에서의 bottleneck 구간 발생과 같은 정보 지연 장애 정도는 심각한 수준에 올라있다. 그러나 현재까지 대부분의 정보 지연 장애의 탐지는 단지 하나 하나의 라우터 만을 검사하는 방법에 그쳤으나, 본 시스템은 네트워크 망 해당 경로 전체를 대상으로 bottleneck 구간 탐지를 할 수 있는 기능을 가지고 있다. 이와 같은 기능에 의해 본 시스템은 웹기반 정보화 사회에서의 네트워크 성능을 높이고 이용 효율을 극대화 할 수 있을 것이며, 또한 여러 분야에서의 기능의 확장으로 매우 유용하게 사용되어 질 것이라 기대된다.

참고 문헌

[1] W.Richard Stevens "TCP/IP Illustrated Volume1 " Addision Wesley 1993

[2] William Stallings "SNMP, SNMPv2, SNMPv3, and RMON1 and 2" 3rd Ed. Addison Wesley

[3] 신상철, 안성진, 정진욱 "SNMP를 이용한 인터넷 분석 파라미터 추출 시스템의 설계 및 구현" 한국정보처리학회 정보처리논문지, p.710-721,제6권 제 3호 1999

[4] Jibiki M, Terano T, Hashida O "Comprehensive bottleneck detection via non-linear optimization techniques" Internet Workshop 1999