웹 브라우저에서 사용자 응답 대기 시간 감소를 위한 설계

이동우°, 조수선
한국전자통신연구원 인터넷정보가전연구부
{hermes, cscho}@etri.re.kr

A Design of Reducing Web Latency on the Web Browser

Dong-Woo Lee°, Su-Sun Cho
Electronics and Telecommunications Research Institute

요 약

인터넷 사용자가 늘면서 정보를 공유 및 서비스하는 웹 서버가 증가하고 있고 그 서비스의 양 또한 엄청난 증가세를 보이고 있다. 이러한 증가로 인한 네트웍의 정체현상은 날로 심각해지고 웹 서버로부터 응답을 기다리는 시간이 점점 증가를 하고 있는 실정이다.

사용자의 응답대기 시간을 줄이기 위해서 서버와 여러 개의 TCP 연결을 고려할 수 있다. 그러나 TCP 연결을 여러 개 하게 되면 네트웍의 트래픽이 증가되는 단점이 있다. 본 논문에서는 이러한 단점을 보완하기 위한 방법으로 과거 웹 서버의 전송속도를 기록하고 이를 이용하여 HTTP 연결 설정을 조정함으로써 네트웍 트래픽의 감소를 꾀하고 자원이 풍부하지 않은 임베디드 시스템을 고려하여 자원 절약 방안을 모색한다. 이 시스템의 장점으로는 클라이언트에 대용량의 캐쉬메모리가 없이도 사용자 응답시간을 줄일 수 있다는 것이다. 임베디드 시스템과 같은 캐쉬 메모리가 부족한 시스템에 적용할 때 효과적인 시스템이다.

1. 서론

초고속 인터넷의 발달로 인터넷을 사용하는 인구의 증가는 기하급수적으로 늘어 나고있다. 웹을 통한 다양한 정보의 공유 및 제공하는 서비스 또한 점점 더 많은 인기를 얻고 있다. 하지만 관련 기술의 발달에도 불구하고 인터넷을 사용하는 사용자들의 수가 늘면서 네트웍 정체현상 및 서버의 과부하는 날로 심각해지고 단말기 앞에서 웹 서버로부터 응답을 기다리는 시간이 증가하고 있다. 사용자들은 웹 서버로부터 더 빠른 응답을 기대하고 있다. 이러한 사용자들의 요구를 수용하기위해서 많은 연구가 이루어졌고 또한 많은 성과가 이루어졌다.

정보가전용 기기는 데스크탑 컴퓨터와는 달리 시스템 자원이 한정되어 있기 때문에 기존의 데스크탑 운영체제와는 다른 요구사항을 가지고 있다. 정보 가전용 기기는 주로 대용량의 하드디스크를 포함하지 않기 때문에 인터넷 브라우저를 위한 큰 용량의 캐쉬 메모리를 지원하지 않는 경우가 대부분이다.

이러한 환경에서 만족스러운 성능을 발휘하기 위해서는 인터넷 접속 시간을 줄이는 문제가 가장 중요하다. 본 논문에서는 이러한 요구를 수용하기 위하여 웹 서버의 과거 전송속도 및 사용중인 HTTP 프로토콜을 고려하여 응답시간을 줄이기 위한 방법을 제시한다.

2. 관련연구

웹 서버로 부터의 응답대기시간에 영향을 미치는 요소에는 DNS 의 IP 변환 시간, TCP 연결 시간, HTTP 요청/응답 시간, 데이터 전송 시간 등이 있다.

웹 서버로부터의 응답시간을 줄이기 위한 많은 연구가 있어 왔고 그 중에서도 캐쉬 및 프락시 시스템을 이용하거나 장애 요소들의 선 처리[1]를 통하여 해결하려는 노력이 꾸준하게 진행되어 왔다.

웹 캐쉬 시스템은 자주 사용되는 문서를 클라이언트 또는 클라이언트 가까이에 있는 시스템에 두는 것으로 웹 서비스의 정체현상을 줄이고 인터넷의 트래픽을 줄여 줄 뿐만 아니라 WWW 시스템의 성능을 향상시킬 수 있는 효과적인 시스템으로 알려져 있다. 캐쉬 시스템의 장점으로는 네트웍 트래픽 및 정체현상 감소, 사용자 응답시간 감소, 웹 서버 부하 감소, 웹 서비스 안정성 증가, 조직이나 기관의 기호 분석 가능 등이 있다. 이러한 장점들이 있는 반면에 적절하게 프락시를 업데이트 시켜주지 않았을 경우에는 최신자료가 아닐 수 있고, 캐쉬서버에 병목현상이 일어날 수 있으며, 캐쉬서버의 오류는 치명적일 수 있다. 또한 서비스하는 웹 사이트의 접속 카운트를 증가 시키지 않아서 서비스 당사자가 자신들의

자료가 캐쉬 서버에 저장되는 것을 원하지 않는 경우도 있다.

캐쉬의 구조로는 캐쉬를 제공하는 서버들 간의 관계에 따른 수직, 분산, 혼합 방식 등의 연구[3][4][5]가 진행되었고 이처럼 여러 개의 캐쉬 서버들 사이의 효율적인 구조의 설계를 통하여 사용자 응답 대기시간을 줄이기 위한 노력도 진행 되고 있다.

사용자 패턴을 분석하여 예측을 통한 선 처리[6]로 사용자 응답대기 시간을 줄이기 위한 노력도 있다. 선 처리는 크게 3 가지로 나누어 지며 웹 브라우저와 웹 서버 사이, 프락시 서버와 웹 서버 사이, 웹 브라우즈와 프락시 서버 사이에서 이루어 진다.

하지만 대용량의 하드디스크를 지원하지 않는 임베디드 시스템의 경우에는 위와 같은 캐쉬 시스템과는 다른 문제해결 방안이 제시되어야 한다. 본 논문에서는 웹 브라우저 자체에서 작은 메모리로 응답대기시간을 줄이기 위한 방법을 제시한다.

3. 웹 서버의 통계적 응답시간을 이용한 연결 설정

본 논문에서는 사용자의 응답대기 시간을 줄이기 위해서 과거 웹 서버의 전송속도를 기록하고 이를 이용하여 HTTP 연결 설정을 조정함으로써 네트웍 트래픽의 감소를 꾀하고 자원이 풍부하지 않은 임베디드 시스템의 효율적인 자원 사용 방안을 모색한다.

본 논문에서는 인터넷상의 특정 웹 서버의 응답시간은 하루 중 시간대에 따라서 실시간으로 변화하지만 웹 서버의 증설 및 네트웍 시설 개선을 하지않는 한 대체로 급격하게 변화를 하지않는다고 가정한다. 따라서 본 연구에서는 새로운 연결 설정을 생성시키는 판단 근거로서 최근 해당 웹 서버의 평균전송속도를 이용한다. 웹 브라우저에서는 웹 서버의 응답시간을 테이블로 관리 하고 과거 웹 서버의 응답 시간을 고려하여 웹 서버와의 연결을 동적으로 조정을 함으로써 응답대기 시간을 줄여 줄 수 있다.

3.1 웹 서버 응답대기 시간 테이블

본 논문에서는 하나의 웹 서버에 대해서 4 개의 항목을 고려하였다.

- HTTP: 웹 서버의 HTTP 프로토콜(HTTP/1.0, HTTP/1.1)
- 접속 시간: 웹 서버에 마지막으로 접속 한 시간
- (TS: Transfer Speed, : bps): 최근 접속 데이터 다운로드 속도
- 평균 전송속도(MTS: Mean of Transfer Speed, 단위: bps): 최근 k 번의 접속시 평균 다운로드 속도

위의 ④번 항목의 경우 n-1 번째에서 최근 k 개를 고려하여 구한 평균 전송속도(MTS_{n-1})에 n 번째의 새로운 전송속도를 추가하여 다시 최근 k 개의 정확한 평균 전송속도(MTS_n)를 계산하기 위해서는 아래의 식으로 계산을 하여야 한다.

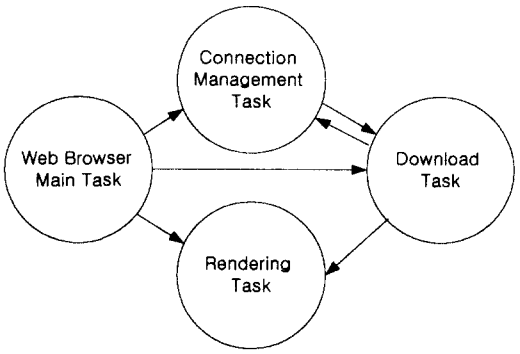
$$MTS_n = MTS_{n-1} + (TS_n - TS_{n-k})/k \quad (k<n)$$

하지만 이 계산방식은 정확한 평균 전송속도를 계산할 수는 있지만 최근 k 개의 전송속도 데이터를 모두 갖고 있어야 하는 문제점이 있기 때문에 아래의 근사 식으로 대체를 한다.

$$MTS_n = MTS_{n-1} + (TS_n - MTS_{n-1})/k \quad (k < n)$$

3.2 웹 브라우저 Task 설계

본 시스템의 웹 브라우저는 크게 Web Browser Task, Connection Management Task, Download Task, Rendering Task 의 4 개 task 로 설계되었다.



(그림 1) 웹 브라우저 Task

그림 1 은 각 task 들 간의 관계를 나타내고 각각의 역할은 다음과 같다.

① Web Browser Main Task:
사용자의 입력 이벤트를 처리하고, HTML 문서를 파싱하면서 문서 내에 서버로부터 전송 받을 데이터가 있을 경우 데이터의 다운로드를 Connection Management Task 에 요구한다. 또 시스템 전반적으로 task 들을 관리한다.

② Connection Management Task:
웹 서버와 효율적인 자원의 이용 및 응답대기 시간 최소화를 고려한 TCP 연결설정에 관련된 전반적인 일을 수행한다. 자세한 것은 다음 절에서 설명한다.

③ Download Task:
Connection Management Task 에서 설정된 연결을 통하여 서버로부터 response message 를 기다리며 전송이 완료되면 Rendering Task 에게 출력을 요구하고 웹 서버 응답대기 시간 테이블을 수정한다.

④ Rendering Task:
Download Task 로부터 출력 요구를 받으면 해당 데이터를 랜더링하여 화면에 출력한다.

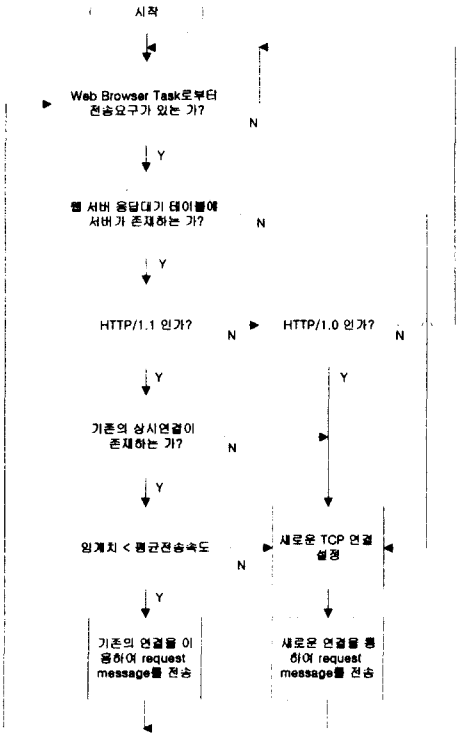
3.3 Connection Management Task 의 동작

Connection Management Task 의 전반적인 동작은 다음과 같다.

① Web Browser Main Task 로부터의 다운로드 요구를 기다린다.

② 다운로드 요구가 왔을 때 웹 서버 응답시간 테이블에서 해당 웹 서버의 데이터를 찾는다.

- 웹 서버의 HTTP 프로토콜 버전 1.0 일 때
새로운 TCP 연결을 설정하고 서버로 request message 를 보내고 Download Task 에 다운로드 요구 메시지를 보낸다.
- 웹 서버의 HTTP 프로토콜 버전 1.1 일 때
해당 웹 서버와 기존의 상시 연결(persistent connection)이 존재하는 지 유무를 찾고 존재하지 않을 경우 새로운 상시 연결을 하여 웹 서버에 request message 를 보내고 Download Task 에게 다운로드 요구를 한다.
기존의 상시 연결이 존재할 경우 웹 서버의 응답대기 테이블에서 평균 전송 속도를 참조하여 일정 임계치 이하이면 새로운 연결을 맺고 그렇지 않을 경우는 기존의 연결을 통하여 request message 를 보낸다.
새로운 연결 설정 판단 기준 식은 아래와 같다.
$TS^{*}w + MTS(1-w) < \text{임계치}$
(w: 최근 전송속도에 대한 가중치, $0 \le w \le 1$)



(그림 2) Connection Management Task 의 동작

또한 기존의 상시연결을 통하여 요구된 데이터의 전송이 완료된 연결이 존재할 경우 임계치를 고려하지 않고 기존의 연결을 재사용함으로써 무분별한 TCP 연결을 통한 네트웍 트래픽 증가 및 시스템 자원의 낭비를 막는다.

응답대기 시간 테이블을 이용한 효율적인 TCP 연결을 통하여 웹 서버로부터의 사용자 응답대기 시간을 줄여줄 뿐만 아니라 시스템의 자원을 절약할 수 있다.

전반적인 Connection Management Task 의 동작을 그림 2 에 나타내었다.

4. 결론 및 향후 과제

본 논문에서는 웹 브라우저 상에서 웹 서버의 응답대기 시간 테이블을 이용한 효율적인 연결설정을 통하여 사용자 응답시간을 줄이기 위한 시스템을 소개하였다.

이 시스템의 장점으로는 클라이언트에 대용량의 캐쉬 메모리가 없이도 사용자 응답시간을 줄일 수 있다는 것이다. 잘 설계된 캐쉬 시스템에서 대용량의 캐쉬메모리를 사용하고도 데이터의 캐쉬 사용률이 약 50%정도인 점을 감안하면 임베디드 시스템과 같은 캐쉬 메모리가 부족한 시스템에 적용할 때 효과적인 시스템이다.

향후 과제는 한국전자통신연구원에서 개발한 Q+ 실시간 OS 기반의 정보가전 용 웹 브라우저에 본 설계를 적용하는 것이다. 또한 임베디드 시스템의 브라우저 특성에 맞는 프락시 서버를 설계하고 웹 브라우저가 이와 연동하여 사용자 응답대기 시간을 줄일 수 있는 노력들이 더 이루어져야 한다.

참고문헌

[1] Jia Wang. "A Survey of Web Caching Schemes for the Internet", ACM Computer Communication Review, 29(5), Oct. 1999.

[2] Edith Cohen, Haim Kaplan. " Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency", In Proceedings of IEEE INFOCOM, Tel-Aviv, Israel, March 2000.

[3] A. Chankhunthod, P. B. Danzig, C.Neerdaels, M. F. Schwartz, and K.J. Worrel, "A hierarchical Internet object cache", Usenix'96, January 1996.

[4] D.Povey and J.Harrison, "A distributed Internet cache", Proceedings of the 20th Australian Computer Science Conference, Sydney, Australia, Feb. 1997

[5] D. Wessels and K. Claffy, Internet cache protocol(IPC), version 2, RFC 2186.

[6] T. Palpanas and A. Mendelzon, "Web prefetching using partial match prediction", Proceedings of WCW'99.