변경 메시지를 이용한 XML 문서의 실시간 갱신

임영환[0] 류기열 위규범
아주대학교
yhlim@netsgo.com, (kryu, kbwee)@ ajou.ac.kr

Real-Time Update for XML Documents using Change Messages

Young-Hwan Lim[0], Ki-Yeol Ryu, Kyu-Bum Wee
Dept. of Information and Communication, Ajou University

요 약

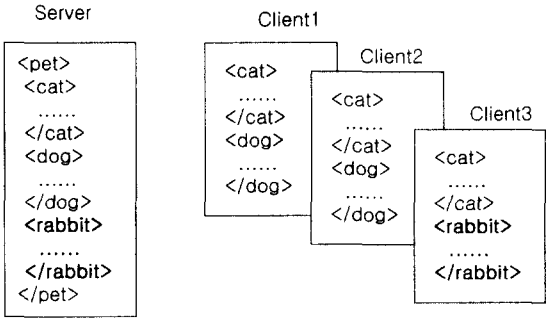
인터넷상에서 문서를 배포하는 데는 기본적으로 HTML의 풀 방식을 사용하며, 간혹 푸시 기술을 이용한 시스템을 이용하기도 한다. 그러나, 이들은 각각 문서의 실시간 변경이라든가 대역폭과 부가 정보에 의한 서버측의 부담 등의 문제를 가지고 있기 때문에 우리는 푸시와 풀 방식을 합쳐 푸시에 의한 부담을 최소한으로 줄이고 문서의 실시간 갱신도 가능하도록 변경 메시지를 이용하였다. 이 변경메시지는 실제 데이터가 아닌 데이터의 변경 정보를 가지고 서버와 연결된 모든 클라이언트에 전달되며 이후 클라이언트에서는 자신이 필요한 경우에만 서버에 데이터를 요청하게 된다. 이러한 구조는 데이터가 변경 되었다는 메시지가 실제 데이터보다 작고, 모든 클라이언트에 데이터를 보내기에 부담이 될 정도로 서버에 연결된 클라이언트가 많은 경우 이전의 다른 시스템들 보다 유용하다.

1. 서론

인터넷에서 문서를 배포하는데 있어서 가장 기본적인 것이 HTTP 프로토콜을 통해 웹 브라우저로 전달되는 HTML문서이다. HTML은 사용자 측에서 서버의 데이터를 가져오는 풀 방식의 서비스이다. 다른 형태의 문서 배포 방식인 CDF(Channel Definition Format)는 XML을 기초로 하여 만들어진 문서 형식으로 HTML 페이지와 다른 웹 리소스 간의 상호 관계를 기술하기위한 메타 데이터 언어를 제공한다. CDF를 채용한 클라이언트는 자동적으로 웹 컨텐츠를 다운로드 하는 "smart-pull" 방식을 사용하고 있다. 그러나 각각은 정보를 가져오는 것이 사용자의 요구나 미리 기술된 주기적인 폴링을 하므로 서버측의 문서 내용이 변경되었을 때, 변경 상황을 즉시 인식하지 못하므로 변경 내용을 반영할 수 없다[1].

또한, 문서의 실시간 변경에 푸시 기술을 이용할 수 있는데, 이 푸시 기술은 웹 브라우저를 통해 인터넷을 헤매는 대신 원하는 정보를 원하는 때에 받을 수 있도록 한다는 데 있어서 각광을 받고 있다. 다시 말해, 푸시 기술은 클라이언트에서 요청이 들어오길 기다리지 않고 서버에서 클라이언트가 받고자 하는 정보가 준비됨과 동시에 자동으로 클라이언트에 전달하도록 되어 있다. 그러나, 서버는 클라이언트 각각이 어떤 정보를 필요로 하는 지에 대한 정보를 가지고 있어야만 하기 때문에, 각각의 클라이언트 측에서 서버 데이터의 일부만을 보유하고 있고, 그런 클라이언트의 수가 매우 많을 경우 서버는 클라이언트측의 정보 유지가 어렵고, 만약 이러한 정보를 유지하지 않는 대신 일률적으로 같은 정보를 전송한다고 한다면 대역폭의 심한 낭비를 초래하게 된다[2].

본 논문에서는 서버측의 문서 변경 정보를 먼저 모든 클라이언트에 전송하고(푸시), 그 후에 클라이언트가 실제 해당 데이터를 가져오는(풀) 혼합된 방식을 통해 XML로 작성된 문서에 실시간으로 정보를 반영 할 수 있는 시스템을 제안하고자 한다.

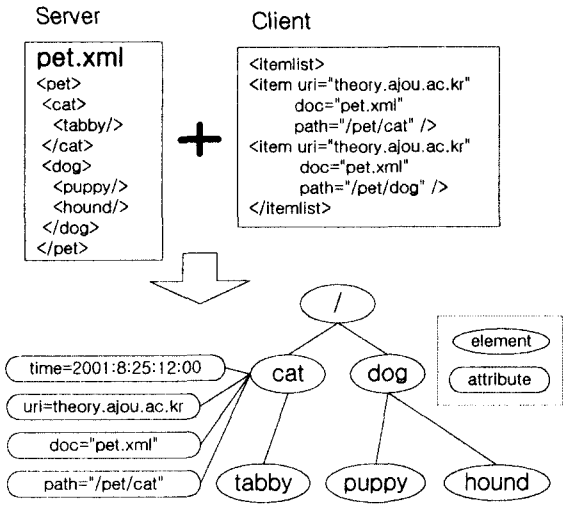


[그림1] 서버와 클라이언트 문서

2. 변경 메시지를 이용한 문서의 갱신

본 논문에서는 클라이언트에 일방적으로 데이터를 전송하는 대신 먼저 서버의 문서 변경에 따라 변경 메시지를 생성 클라이언트에 전송하고, 클라이언트는 이때 해당하는 정보를 다시 요청하는 방식을 이용한다. 이를 위해서 클라이언트는 서버에서 자신이 필요로 하는 데이터의 정보를 기술한 아이템 리스트 파일을 로컬 영역에 유지한다.

변경 메시지의 장점은 서버 문서 데이터의 변경 상태를 클라이언트가 인식하게 하고, 되도록 적은 양의 데이터만 네트웍을 통해 전달되도록 할 수 있다는 데 있다. [그림1]에서 보여지는 것처럼 클라이언트들이 서버의 데이터를 참조하고 있다고 할 때, 'rabbit'의 정보는 Client3에서만 참조하고 있기 때문에 모든 클라이언트에 이 데이터를 일방적으로 전송하는 대신 먼저 'rabbit' 데이터가 변경됨과 동시에 서버는 'rabbit' 데이터가 새로운 내용으로 수정되었다는 메시지만을 전송한다. 이 메시지는 실제 내용 데이터보다 상대적으로 매우 작으므로 대역폭의 낭비는 크지 않다. 그리고, 나서 이 메시지를 받은 클라이언트 중 'rabbit'데이터를 참조하고 있는 클라이언트 3번만 데이터를 요청하고 서버로부터 데이터를 받아오게 된다.



[그림2] 아이템 리스트와 클라이언트 문서의 구조 트리

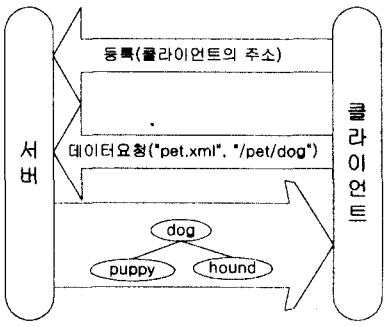
2.1 아이템 리스트와 클라이언트 문서

아이템 리스트는 클라이언트가 서버측의 데이터를 가져오기 위한 정보를 담은 XML 어플리케이션이다. 이 아이템 리스트는 <item> 엘레먼트와 각종 필요한 애트리뷰트로 기술한다. 애트리뷰트 중에서 'uri'는 해당 문서가 존재하는 서버의 주소이며, 'doc'는 클라이언트가 관심을 가지고있는 서버의 문서파일의 이름, 'path'는 클라이언트가 관심을 가지고 있는 엘레먼트를 가리킨다. 특히 'path'는 XPath[4]의 서브 셋으로 문서 내에서 아이템을 식별하는데 매우 중요한 요소이다. [그림2]의 우측 상단에서 그 예를 볼 수 있다.

또한, 부가적으로 XPath로 기술된 엘레멘트로부터 몇 개의 엘레먼트까지 가져올지를 기술하는 'quantity', XPath로 기술된 엘레먼트로부터 몇 단계 아래의 자식 노드들을 지 가져올지를 결정하기위한 'depth', 그리고, 데이터의 분석을 목적으로 정보가 변경되어가는 과정을 클라이언트에 유지하려 할 때 기술하는 'history'등 많은 다양한 애트리뷰트들이 사용 될 수 있다[3].

클라이언트는 이와 같이 제시된 아이템 리스트를 읽고 나서 서버에 자신의 주소를 등록하고, 참조하고 있는 문서의 이름과 xpath를 보내면, 서버에서는 해당하는 데이터를 클라이언트로 전송한다. 이때, 클라이언트에서는 전송 받은 데이터와 아이템 리스트의 정보를 합쳐 [그림2]의 하단에서 보이는 바와 같이 새로운 XML DOM[5] 트리를 생성한다. 여기에 메시지의 중복이나 오류에 의해 데이터를 잘못 업데이트 하는 것을 막기위한 시간 정보 등 조작에 필요한 부가정보를 추가한다.

2.2 변경 메시지



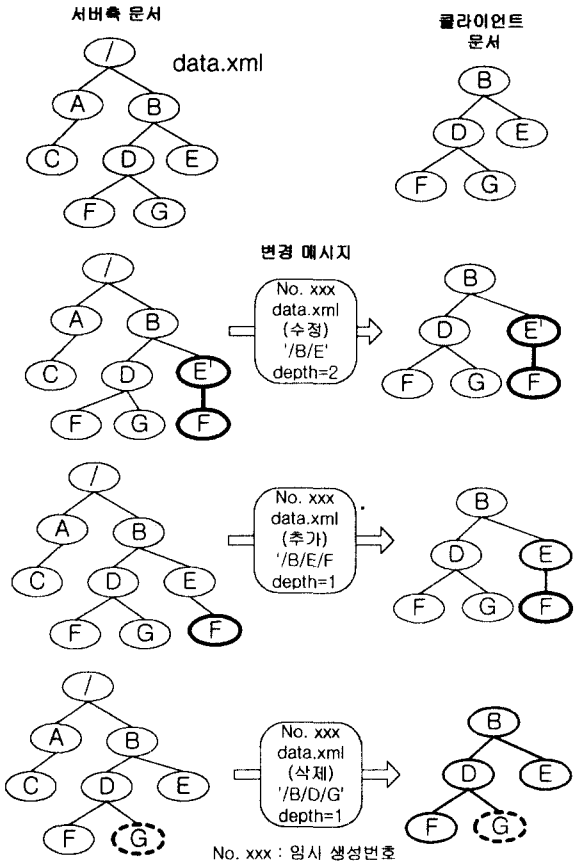
[그림3] 아이템 리스트 참조

변경 메시지는 서버내의 문서 내용이 변경되었을 때, 등록된 모든 클라이언트로 전송되는 메시지다. 이 메시지는 임시번호, 메시지 종류, 관련문서이름, path, 변경값이를 기본 정보로 갖는다[그림4].

임시번호는 적당한 주기안에서 데이터의 변경이 이루어질 때 고유하게 할당되는 번호이며, 변경 메시지별로 고유한 값을 가지도록 생성된다 클라이언트 측에서 이 변경메시지를 보고 데이터를 요청할 때, 이 고유 번호를 다시 보내온다. 이러한 과정은 서버측에서 클라이언트가 어떤 메시지를 보고 데이터를 요청해 온 것인지 확인할 수 있으므로 클라이언트에서 요청 메시지를 보낼 때 변경 메시지에 담겨진 내용들을 다시 요청 메시지에 담아 보낼 필요가 없도록 하고, 메시지 간의 충돌이나 중복 등의 문제를 해결 하게 한다.

메시지 종류는 수정, 추가, 삭제로, 수정 메시지라면 갱신이 필요한 데이터를 확인하고 서버에 요청하며, 추가, 삭제 메시지는 클라이언트 자체에서 완료된다.

관련문서는 정보가 변경된 서버의 문서 이름을 가리키며, 문서의 변경된 최상의 엘레먼트의 XPath인 path, 변경값이와 함께 클라이언트에서 변경된 데이터를 식별하고, 이를 처리할지 말지를 결정하는데 이용된다. 변경범위는 변경된 범위가 해당 엘레먼트에만 해당하는 경우 신규데이터를 전송할 때, 필요 없이 자식 노드까지 포함해서 전송되지 않게 하고, 또한, 클라이언트의 XML 트리의 depth 애트리뷰트와 비교하여, 변경 데이터가 클라이언트 depth의 아래인 경우 메시지를 무시할 수 있도록 한다. XML 문서를 대상으로 하기 때문에 한 엘레먼트 내에 동일한 엘레먼트 이름을 가지는 경우가 발생할 수 있으나, 이 역시 'position()' 등 XPath의 각종 함수들을 이용하여 구분할 수 있다.



[그림4] 변경 메시지

3. 분석

[표1]은 제안된 시스템과 일반 HTML을 받는 풀, CDF가 이용하고 있는 smart-pull, 서버가 클라이언트가 요구하는 정보를 유지하고 전송하는 푸시 방식, 그리고, 모든 클라이언트에 동일한 데이터를 일률적으로 전달하는 TV와 같은 방식의 푸시 방식을 비교한 것이다.

HTML과 CDF의 'smart-pull' 방식은 사용자의 요청이나 미리 결정된 주기에 따라 데이터를 가져오기 때문에 모든 방식 중에서 가장 오버헤드가 적다. 그러나, 서버 데이터에 변경이 가해졌을 때, 변화를 알 수가 없기 때문에 사용자에게 항상 최신의 데이터를 보여줄 수가 없다.

푸시 방식 중, 클라이언트가 필요로 하는 정보를 유지하는 시스템의 경우 서버는 정보가 변경 되었을 때, 이 데이터에 관련한 클라이언트에만 정보를 보내줄 수 있다. 그러나, 클라이언트가 수시로 서버에 연결하고, 끊어지는 경우 클라이언트의 정보를 등록하고, 유지하는 과정이 심한 부담이 될 수 있다. 또한, 일방적으로 동일한 데이터를 보내는 경우 많은 클라이언트가 전혀 쓸모없는 다량의 데이터를 받게 되기 때문에 상당한 대역폭의 낭비가 생긴다.

반면, 제안하고 있는 방식은 변경 정보를 통해 문서 변경에 대한 실시간 정보를 인식할 수 있으며, 비록 변경 정보는 모든 클라이언트에 전송되지만, 그 크기가 상대적으로 매우 작기 때문에 변경 메시지의 전송은 거의 부담을 주지 않는다.

[표1] 문서 배포 방식들의 비교

	풀		푸시		제안 시스템
	풀 (HTML)	Smart -Pull (CDF)	클라이언트의 등록	일률적 전송	
실시간 인식	불가능	불가능	가능	가능	**가능**
서버측의 클라이언트 정보 유지	불필요	불필요	많다	적다	**적다**
대역폭 낭비	없다	없다	적다	많다	**적다**
클라이언트 대기상태	안함	안함	필요	필요	**필요**
신규데이터 갱신 시점	사용자의 요청 시	주기적	변경 즉시	변경 즉시	**변경 즉시**

4. 결론 및 향후계획

본 논문에서 제시하고 있는 기법이 유용하기 위해서는 다음의 몇 가지 가정을 세워야 한다. 첫째로, 데이터가 변경되었다는 메시지는 실제 데이터에 비해 매우 작아야 한다. 둘째, 모든 클라이언트에 데이터를 보내기에 부담이 될 만큼 많은 클라이언트가 서버에 연결 되어 있어야 한다. 셋째, 클라이언트가 항상 대기(idle)상태로 있는 것이 시스템에 큰 부담을 주지 않아야 한다.

위와 같은 가정이 만족되는 환경이라면, 제안하고 있는 시스템은 실시간으로 항상 새로운 문서를 제공하면서도 불필요한 데이터의 전송을 최소한을 줄일 수 있다.

그러나, 제안 하고 있는 시스템은 위에서 가정하고 있는 특정 환경 하에서는 충분히 쓸모가 있다고 할 수 있지만, 실제 서비스화 되기 위해서는 여러 가지 문제점들을 해결해야만 한다. 예를 들어, 최초 클라이언트 문서를 작성하기 위해 서버에서 제공하고 있는 문서들의 정보를 어떻게 제공하고 클라이언트는 이 정보를 어떻게 이용할 것인가, 문서의 데이터가 아니라 문서의 구조자체가 변경된 경우 변경 정보를 주고 받는 데 있어서 DTD 관련 정보의 교환에 대해서는 어떻게 처리할 수 있는가 등이다. 또한, 이렇게 제안된 시스템이 구체적으로 어떤 서비스에 이용될 수 있을 지에 대하여도 좀더 많은 연구가 필요하다.

참고문헌

[1] A. Hoff, H. Patrovi, "The Open Software Description Format (OSD)", Microsoft Corp. http://www.w3.org/TR/NODE-OSD.html

[2] H. Mei, "Performance Enhancement of Web-Based Push Transmitter with Channel Scheduling", Proceedings of the The Twenty-Fourth Annual International Computer Software and Applications Conference, p377-382, 2000.

[3] D. Hunter, "Beginning XML", p.2-20, Wrox, 2001.

[4] J. Clark, S. DeRose, "XML Path Language (XPath) Ver. 1.0", W3C Recommendation, November 1999.

[5] T. Pixley, "Document Object Model (DOM) Level 3 Core Specification Ver. 1.0", W3C Working Draft, June 2001.