자원 제약적 클라이언트에서 XML 스트림의 안전한 처리 기법

안은주° 박 석
서강대학교 컴퓨터공학과
{dksdw°, spark}@dblab.sogang.ac.kr

A Secure Processing of XML Stream at Resource Limited Client

Eunju An° Seog Park
Department of Computer Science , Sogang University

요 약

기존 서버에서 담당하던 접근제어를 서버에 비해 제약을 갖는 클라이언트에서 처리하고자 하는 요구가 생겨남에 따라 자원 제약을 갖는 환경에서 효율적으로 처리할 수 있는 가벼운 접근제어 방법이 필요하게 되었다. 기존의 접근제어 연구는 안전성에 초점을 맞추어 왔기 때문에 효율성 측면에 대한 고려가 적었으며, 최근 스트림 환경에서 보안 문제가 대두되면서 접근제어를 포함한 보안 측면의 연구가 시작되었으나 시작 단계에 불과하다. 본 논문은 이러한 환경의 변화를 배경으로 최근 데이터 저장과 전송의 표준으로 부각되고 있는 XML 문서의 데이터 스트림을 PDA나 휴대용 단말기와 같이 자원의 제약이 있는 클라이언트에서 안전하고 효율적으로 다루기 위한 방법을 제안한다. 첫째로 한정된 메모리 내에서 안전한 결과를 내기 위해 오버헤드가 매우 적은 접근제어 처리 방법을 제안하고 있으며, 둘째로 접근제어 추가로 인한 오버헤드를 상쇄시키기 위해 처리 단계 마다 최적화가 가능한 부분들을 찾아 성능을 개선하고자 하였다.

1. 서 론

기존의 접근제어 환경에서는 신뢰성 있고 성능이 뛰어난 서버가 개개의 사용자에 맞는 정보를 제공해주는 역할을 담당했다. 그러나 여러 가지 요인[1]들로 인해 서버에서 담당했던 접근제어 프로세스를 클라이언트에서도 담당할 필요가 생겨나고 있다. 실제로 Windows Media Player[2]는 개인 PC에서 소프트웨어 적으로 출판된 자산을 보호할 수 있도록 하는 솔루션을 제공하고 있으며, PC나 PDA등에서 보안 토큰이나 스마트카드 등을 탑재함으로써 하드웨어적으로 보안 기능을 제공하는 경우도 있다. 그러나 이러한 보안 기능은 거의 데이터 암호화에 의지하고 있으며 사용자 마다 다르게 접근할 수 있는 해결책으로 부족함이 있다.

본 연구는 이러한 환경의 변화를 배경으로 XML 문서의 데이터 스트림을 PDA나 휴대용 단말기와 같이 자원의 제약이 있는 클라이언트에서 세밀한 접근제어 수행함으로써 안전하고 효율적으로 다루기 위한 방법을 제안한다.

본 논문의 연구동기 예로 다음과 같은 상황을 설정해 보았다. u-헬스 시대를 맞아 의사나 환자들이 휴대용 단말기를 통해 정보를 주고받는 것이 가능해졌다. 이러한 환경에서 환자들은 자신의 정보를 병원 정보 시스템에 등록할 것이고 의사들은 이를 통해 환자의 정보를 조회할 수 있다. 이 때 의사들은 담당환자의 질병에 관한 꼭 필요한 정보를 제외하고는 볼 수 없어야 한다. 그러나 서버는 정보를 제공받는 클라이언트 모두에게 구별된 데이터를 전송해 주지 못한다. 이는 다양한 사용자에 맞는 접근제어 규칙을 서버에서 모두 관리하기 어려우며, 환자의 상태를 실시간으로 전송하기 위해 데이터를 지속적으로 보내야 하는 스트림 환경에서 이러한 여러 가지 경우의 결과를 모두 브로드캐스트(broadcast) 하는데 상당한 비용이 소비되기 때문이다. 따라서 서버는 모든 정보를 브로드캐스트 하지만 클라이언트에서 각자 접근이 허가된 내용만을 읽어 사용자의 질의에 응답할 수 있어야 한다.

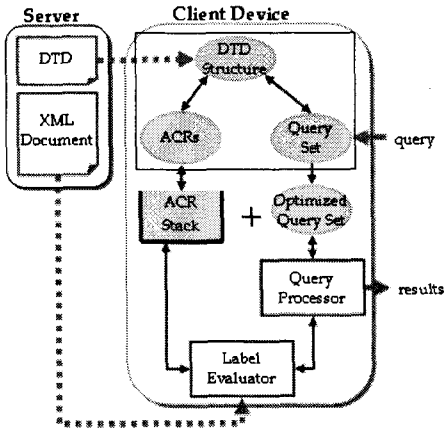
2. 관련연구

XML 접근제어 연구로는 크게 뷰 기반 접근제어 모델(View-based Access Control Model:VACM)과 질의 재작성 모델이 있다. 뷰 기반 접근제어 모델은 보안 관리자가 각 사용자의 접근제어 규칙에 맞게 XML 문서에 대한 뷰를 생성하는 방법이다[3]. 즉, 사용자의 접근제어 규칙에 의해 접근 가능한 노드만 유지하고 접근 불허된 노드는 보여주지 않음으로써 실제로 접근 가능한 노드에 대해서만 질의 계산을 수행하도록 한다. 이러한 모델은

여러 사용자에 해당하는 뷰를 모두 유지해야 하므로 높은 유지비용과 저장비용의 단점을 가지고 있다. 때문에 이러한 단점을 해결하고자 등장한 연구가 질의를 재작성하여 안전한 질의를 생성하는 방법[4][5]이다. 기존의 XML 접근제어 연구들이 안전한 결과를 도출하는 데에만 초점을 맞췄다면 SQ-Filter 시스템은 질의와 관련된 접근제어 정책만을 찾아내어 꼭 필요한 접근제어 정책만을 처리하도록 함으로써 시스템의 효율성 측면을 함께 고려한 질의 재작성 모델이다. 트리 구조에 Preorder와 Postorder 값을 부여하는 방법은 XML 문서를 트리 구조로 만들어 XML 질의 언어를 위한 질의 처리기에서 유용하게 사용하는 방법으로 XML 데이터에 대한 전통적인 접근제어 메커니즘은 질의를 요구한 사용자의 모든 접근제어 규칙들을 이용하는 반면 SQ-Filter 시스템은 접근제어 규칙의 PRE(order), POST(order) 값과 질의의 (PRE, POST) 값을 이차 평면 (이를 PRE/POST 평면 이라 함)에 사상시켜 그 질의에 관련된 조상(ancestor) (혹은 부모(parent)) 및 자손(descendant) (혹은 자식(child)) 관계에 있는 접근제어 규칙들만을 이용한다. 결국, 사용자의 접근제어 규칙들 중 질의에 해당되는 최소의 접근제어 규칙들을 이용하기 때문에 처리량이 줄어들고 빠른 질의 재작성이 가능하다는 장점을 가지고 있다. 그러나 이러한 질의 재작성은 복잡한 질의를 생성하고 모든 재작성 된 질의에 대해 정확한 결과를 얻기 힘들다는 단점을 가진다. [7]의 클라이언트 기반 XML 문서 스트림 접근제어 연구(Client-based access control management for XML documents)는 질의 재작성을 피하고 접근제어 규칙과 질의 계산을 동시에 수행함으로써 현재 입력된 데이터에 대한 접근 여부를 판단한다. XML 접근제어는 질의와 마찬가지로 접근제어 규칙도 XPath 경로 표현식[7]으로 나타내고 있다. 따라서 접근제어 규칙의 계산 또한 질의 계산과 유사한 방법으로 처리할 수 있다. [7]의 연구는 XML 문서에 대한 빠른 질의 처리 방법인 Yfilter[8] 방법을 접근제어 규칙에도 적용함으로써 빠른 접근제어 규칙 계산을 가능하게 하였다. [7]의 연구는 이렇게 접근제어 규칙을 오토마타(Access Rules Automata:ARA)로 변환하되 긍정적 접근제어 규칙에 대한 오토마타와 부정적 접근제어 규칙에 대한 오토마타를 따로 유지한다. 접근제어 규칙 오토마타 계산을 통해 최종 상태에 이른 규칙이 있으면 이를 접근제어 스택에 push 함으로써 현재 접근 가능 여부를 판단한다. [7]의 연구는 접근제어 스택의 상태를 통해 입력되는 데이터의 접근 여부를 판단하므로 질의를 접근제어 규칙에 맞추어 재작성 할 필요가 없으며, 접근이 불허된 데이터에 대한 접근이 불가능하므로 안전한 질의 결과를 내보내게 된다. 또한 빠른 질의 처리 방법인 Yfilter 방법을 접근제어 처리에 이용함으로써 빠른 접근제어 처리를 가능하게 하고 있다. 그러나 이 방법은 질의 처리와 접근제어 처리를 서로 다른 프로세스로 보고 있으므로 두 개의 유사한 프로세스 처리를 위해 중복되는 과정을 반복한다.

3. 안전한 결과를 내는 XML 데이터 스트림의 가벼운 처리 방법

3.1 시스템의 구조



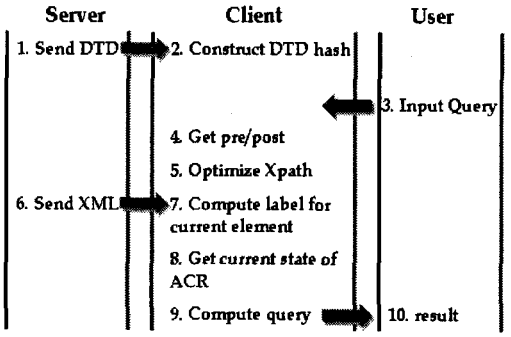
[그림 1] 제안하는 시스템의 구조

[그림 1]은 본 연구에서 제안하는 시스템의 구조를 보여주고 있다.

서버; 서버는 수많은 데이터 소스들로부터 정보를 제공받고 이들 정보를 취합하여 하나의 XML 문서로 만든다. 따라서 클라이언트에 브로드캐스트 할 DTD 문서와 XML 문서를 가지고 있으며 이들 문서가 변경될 때 마다 클라이언트에 정보를 보낸다.

클라이언트; 클라이언트는 서버로부터 전송 받은 DTD를 저장하며, 접근제어 규칙 및 사용자로부터 입력 받은 질의, 그리고 끊임없이 입력되는 XML 파일을 처리하기 위한 모듈들을 가지고 있다. 본 연구에서는 휴대용 단말기와 같이 메모리 자원이 적은 클라이언트를 가정하고 있다.

시스템에서 XML 문서가 처리되는 과정은 간략하게 전처리(pre-processing) 과정과 질의 처리(query-processing) 과정으로 나눌 수 있다.



[그림 2] XML 문서 스트림 처리 과정

[그림 2]는 XML 문서 처리를 위한 절차를 보여주고 있다. DTD나 사용자 입력 질의는 자주 바뀌지 않으므로 전처리 과정을 통해 이러한 정보가 한 번 등록되고 나면 DTD 혹은 질의가 변경될 때에만 전처리 과정이 수행된다. 따라서 이러한 경우를 제외하고는 질의 처리 과정만 수행된다.

3.2 전처리(pre-processing) 과정

3.2.1 DTD 해시 구성

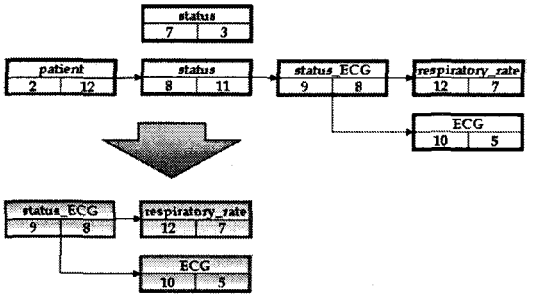
서버가 DTD 문서를 전송하면 클라이언트는 질의 계산 및 접근제어 규칙 계산에 이를 활용하기 위해 DTD를 순차적으로 접근하며 엘리먼트 정보를 Pre/Post 값으로 변환하여 DTD 해시를 구성한다. DTD 해시는 태그 이름(tag name)을 키로 가지며 네 개의 정보({tag name, pre, post, parent})를 유지한다. SQ-filter의 경우 질의와 관련된 접근제어 규칙을 찾기 위해 위의 DTD 해시가 이용되며, [6]의 경우 질의나 접근제어 규칙 계산을 위해 DTD의 정보를 이용하지 않는다. 본 연구에서 DTD 해시를 사용하는 이유는 첫째, 질의나 접근제어 규칙 계산 시 이러한 메타 정보를 유지함으로써 보다 빠르고 정확하게 계산하기 위함이며 둘째, 질의와 접근제어 규칙의 계산은 유사하므로 DTD 해시 정보를 질의 계산과 접근제어 규칙 계산에 공유할 수 있기 때문이다.

3.2.2 질의 및 접근제어 규칙 등록

질의와 접근제어 규칙은 모두 XPath 표현식으로 나타내므로 앞에서도 언급했듯이 이들의 처리는 유사하게 진행된다. XPath 경로에 해당하는 Pre/Post 값을 탐색하는 과정은 다음과 같다. 탐색의 순서는 목적 노드로부터 경로 표현식의 시작 노드로 거슬러 올라간다. 이 때 프레디키트의 분기 노드를 만나면 프레디키트의 값을 먼저 탐색한 후, 다시 분기 노드의 이전 노드에 대한 탐색을 계속 한다. 이 때, 탐색된 값이 XPath를 만족하는지 알기 위하여 먼저 탐색된 노드의 (pre, post) 값과 새로이 탐색된 노드의 (pre, post) 값을 비교하여 새로이 탐색된 노드가 조상 관계에 있는 노드인지 확인한다. 이 과정을 거치면 DTD 형식에 맞지 않는 표현식은 걸러지고 올바른 표현식만 남게 된다.

Pre/Post 구조에서는 임의의 노드에 대한 (pre, post) 값을 알면 DTD 상의 정확한 위치를 파악할 수 있다. 이는 곧 해당 노드에 대한 루트(root) 부터의 정확한 경로를 알 수 있음을 뜻한다. 따라서 XPath 경로 표현식에 나타나는 모든 노드의 정보를 유지하는 대신 몇 개의 필요한 노드에 대한 정보만 유지하여도 XPath 경로를 알아낼 수 있다. 따라서 최적화 과정을 거치게 되는데, 최적화 과정에서는 질의의 목적노드, 프레디키트 경로의 단말노드, 그리고 프레디키트의 단말 노드와 질의의 목적 노드가 동일한 질의에 속해 있다는 관계를 파악하기 위해 질의의 분기 노드에 대한 (pre, post) 정보만 유지하도록 한다.

위의 과정을 거쳐 등록된 질의와 접근제어 규칙들은 Pre/Post 정보를 바탕으로 질의 계산 시 입력 데이터와 빠른 비교가 가능하며, 최적화를 통해 질의 및 접근제어 규칙 유지에 필요한 메모리 사용량과 질의 계산 시 필요한 비교 횟수를 줄일 수 있다.



[그림 3] Pre/Post 변환 질의와 최적화 질의

P1:
//patient/status/status_ECG[ECG>50]/respiratory_rate

[그림 3]의 자료구조는 위의 XPath 경로를 DTD 해시 정보를 이용해 변환한 상태와 이를 최적화 한 상태를 보여주고 있다.

3.3 질의 처리 과정

질의 처리 과정에서는 전처리 과정에서 등록된 정보를 이용하여 접근제어 처리와 질의 처리를 동시에 수행한다.

3.3.1 입력 데이터 정보 계산

서버로부터 XML 문서를 전송 받으면 질의와 접근제어 규칙 계산을 위해 입력된 데이터의 (pre, post) 정보를 알아야 한다. DTD 해시는 입력된 데이터의 태그 이름으로 알맞은 (pre, post) 값을 빠르게 탐색할 수 있게 해준다. 그러나 별도의 정보를 유지하지 않으면 여러 개의 (pre, post) 값이 탐색될 경우, 입력 데이터에 대한 DTD 상의 정확한 위치를 알 수 없게 되므로 알맞은 (pre, post) 값을 결정하기 위해 질의 경로 상에 나타나는 모든 노드를 유지하고 있어야 하고 전처리 과정에서 설명했던 질의 최적화가 불가능해진다. 따라서 본 연구에서는 새로이 입력될 데이터의 정확한 정보를 알아내기 위해 이전에 입력된 데이터의 정보를 유지하도록 한다. 새로운 데이터가 입력되면 바로 이전에 입력된 데이터의 정보를 이용하여 (pre, post) 값을 계산한다. 이 방법은 입력된 데이터의 정보 계산을 위해 이전에 입력 받은 데이터의 (pre, post) 정보만을 유지함으로써 질의 최적화를 가능하게 하므로 질의 경로 상의 모든 노드를 유지하는 것 보다 메모리 면에서 효율적이다.

입력 데이터의 정보는 두 가지 경우로 나누어 계산된다. 입력 데이터 계산을 위해 유지해야 할 정보를 Current_data라 하자.

먼저 엘리먼트의 시작(start element)을 알리는 태그가 입력되면 이는 바로 이전 데이터의 자식이거나 루트 노드이다. Current_data의 값이 NULL 이면 입력된 데이터는 루트 노드이고, DTD 해시에서 루트 노드에 대한 정보를 쉽게 찾을 수 있다. 입력 데이터가 루트 노드가 아닌 경우, DTD 해시에서 입력된 데이터의 태그 이름으로 (pre, post) 값을 탐색한다. 찾은 결과 중 기존에 유지하고 있던 데이터의 자식에 해당하는 값을 찾으면 Current_data의 값을 새로 찾은 값으로 바꾼다. 엘리먼트의 끝(end element)을 알리는 태그가 입력되면 다음에 입력될 데이터를 위해 Current_data의 값을 이전 값으로 돌려놓는다.

이렇게 Current_data를 유지하면 질의 계산 시 빠르고 정확한 비교를 가능하게 하며 질의의 모든 정보를 유지할 필요가 없으므로 질의 최적화를 통한 가벼운 처리가 가능해 진다.

3.3.2 접근제어 규칙 계산

질의 계산과 접근제어 규칙의 계산은 동일하나 입력되는 데이터의 접근 가능 여부를 판단하기 위해 접근제어 규칙을 계산 할 때에는 접근제어 스택을 유지한다. 접근제어 규칙이 명시된 엘리먼트가 입력될 때 마다 해당 접근제어 규칙의 접근 가능 여부를 스택에 push하게 되는데, '전파허용' 정책에 의해 입력되는 데이터의 접근 가능 여부는 스택의 top에 해당하는 값을 기준으로 판단하게 된다.

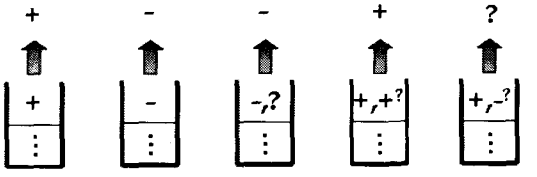
접근제어 스택은 [7]과 같이 (Positive Active, Positive Pending, Negative Active, Negative Pending)의 네 가지 값을 가질 수 있다. Active 상태는 프레디키트를 포함한 접근제어 규칙을 만족하여 해당 규칙의 접근 여부가 적용되는 상태이며, Pending 상태는 프레디키트를 제외한 규칙을 만족하지만 프레디키트의 조건을 만족하는지 따져본 후 접근 여부를 적용할 수 있는 상태를 말한다. 본 연구에서 사용하는 접근제어 모델에 맞추어 접근제어 스택의 계산 방법을 정의하면 다음과 같다.

- 본 연구는 '닫힌 정책'에 따라 스택이 비어있을 경우, 기본적으로 데이터에 접근할 수 없다. 따라서 질의를 만족하더라도 스택의 top에 Positive Active 값이 오기 전까지 결과를 내보내지 않는다.
- 스택의 top에 Positive Active 값이 있는 경우, 입력 데이터에 대한 접근이 허용되고 질의를 계산하여 만족하면 결과를 내보낸다.
- 스택의 top에 Negative Active 값이 있는 경우,

본 연구에서 사용하는 접근제어 모델은 효율성을 고려해 '접근 불허 하향 일관성' 정책을 따르므로 접근이 불허된 노드의 하위 노드는 접근할 수 없다. 따라서 이후에 push 되는 값들은 모두 Negative Active로 변환한다. 이 경우 입력 데이터에 대한 접근이 허용되지 않으므로 질의를 계산하지 않는다. 즉, 질의 입장에서는 데이터가 입력되지 않은 것처럼 처리하게 된다.

- 스택의 top에 Positive Pending 값이 있는 경우, 이는 잠재적으로 접근 가능할 수 있으므로 만약 질의를 만족하면 그 결과를 임시로 저장한다. 만약 나중에 프레디키트를 만족하면 Positive Active로 값이 바뀌고 임시로 저장했던 질의 결과를 사용자에게 내보낸다. 프레디키트를 만족하지 않으면 스택에서 pop 하고 임시로 저장했던 질의 결과는 버린다.
- 스택의 top에 Negative Pendig 값이 있는 경우, 이는 잠재적으로 접근 불가능하거나 아예 접근제어 규칙을 만족하지 않을 수도 있다. 만약 이후에 프레디키트를 만족하게 되면 Negative Active로 바뀌고, 이 경우 스택에서 상단에 push 되었던 값들도 모두 Negative Active로 변

경된다. 상단에 Positive Pending 값이 있을 경우 그 값이 Negative Active로 바뀌면서 임시로 저장했던 질의 결과는 사용자가 접근할 수 없는 정보가 되므로 버리게 된다.



[그림 4] 접근제어 스택의 계산

제안 방법의 질의 처리 과정은 질의 계산이 간단하여 빠른 질의 계산이 가능하다. 또한 접근제어 규칙 적용을 위해 사용되는 접근제어 스택의 경우 최대 접근제어 규칙의 수만큼 데이터를 가지게 되며, 임의의 시점에서 보면 서버로부터 임의의 데이터가 입력되었을 때 DTD 트리 상에서 입력된 노드의 조상 노드들 중 접근제어 규칙이 적용된 노드의 수만큼의 데이터를 가진다. 클라이언트 환경에서 접근제어 규칙의 수가 많지 않음을 고려하면 접근제어 스택이 차지하는 공간은 매우 적다.

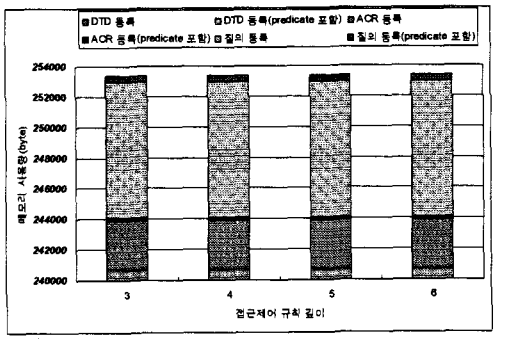
4. 실험 및 분석

본 실험에서 제안하는 기법은 자바로 구현하였으며, 펜티엄 IV 3.0Ghz 프로세서와 1GB DDR2 메모리가 장착된 마이크로소프트 윈도우 XP 운영체제에서 자바 가상머신 1.5.0을 이용하였다. 실험을 위한 입력 문서로는 SigmodRecord.xml(467Kb) 파일을 사용하였으며, 접근제어 규칙과 질의의 수는 클라이언트 환경임을 고려해 최대 10개를 넘지 않도록 하였다.

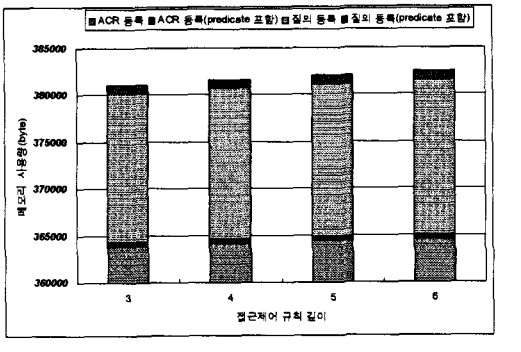
4.1 기존 연구와의 비교

[그림 5]와 [그림 6]은 각각 제안 방법과 관련연구[7]의 전처리 과정에서 소모되는 메모리 사용량을 비교하였다. [그림 5]와 [그림6]의 차이점은 앞의 경우는 규칙의 깊이와 상관없이 동일한 메모리를 사용하였으나 후자의 경우는 규칙의 깊이가 깊어질수록 메모리 사용량이 증가하였다는 점이다. 제안 방법은 질의와 규칙의 등록 시 최적화 과정을 거치므로 질의나 규칙의 깊이가 길어지더라도 이에 따른 메모리 사용량의 추가 비용이 발생하지 않는다. 또한 [그림 5]의 전처리 과정에서 사용된 메모리가 255Kb를 넘지 않는 반면 [그림 6]의 전처리 과정에서 사용된 메모리가 380Kb 이상임을 통해 제안 방법의 경우 최적화를 통해 질의 및 접근제어 규칙 저장을 위해 필요한 메모리를 줄였음을 확인할 수 있다.

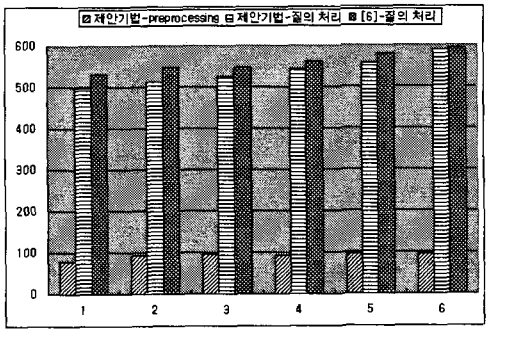
[그림 7]은 두 기법의 접근제어 규칙 처리를 포함한 질의 처리 시간을 접근제어 규칙의 수를 변화시켜 가며 비교한 결과인데, 전체적으로 제안 방법의 성능이 우수함을 확인할 수 있다.



[그림 5] 규칙 깊이 별 메모리 사용량(제안기법)



[그림 6] 규칙 깊이 별 메모리 사용량(관련연구[6])

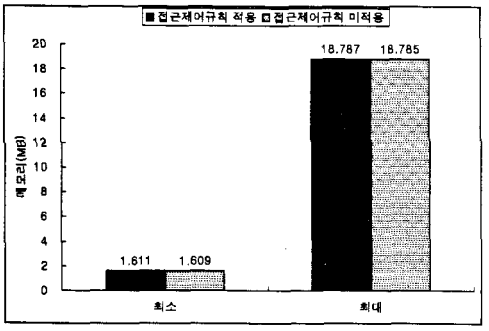


[그림 7] 규칙 개수 별 처리시간

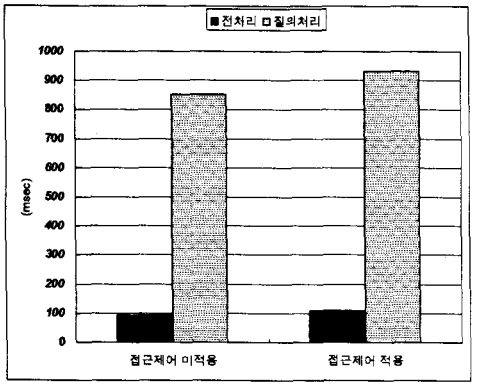
4.2 접근제어 적용 추가 비용

[그림 8]은 제안 기법의 접근제어 규칙 적용 시와 미적용 시의 메모리 사용량을 보여주고 있다. 그림에서 확인할 수 있듯이 최소와 최대 모두 두 경우의 메모리 사용량 차이가 2KB 정도임을 알 수 있다. [그림 9]의 결과를 보면 전처리 과정의 경우 평균적으로 접근제어 적용 전과 후가 15msec 정도 차이를 가지며 질의 처리 과정의 경우 평균 100msec 보다 적은 차이를 보이고 있다.

실험 결과에서 확인하였듯이 접근제어 적용으로 인한 추가 비용이 매우 적은 이유는 우선 전처리 시 접근제어 규칙 등록 시 사용하는 pre/post 구조가 질의 등록 시 사용하는 DTD 해시를 공유하여 사용하므로 접근제어를 위한 추가 속성 정보를 필요로 하지 않으며, 접근제어 규칙 등록 시 최적화 과정을 거침으로써 유지해야 하는 접근제어 규칙 정보의 크기를 줄였고, 이로 인해 질의 처리 과정에서 접근제어 규칙 계산 시 비교할 노드의 수를 줄였기 때문이다.



[그림 8] 평균 메모리 사용량



[그림 9] 평균 처리 시간

5. 결론

본 논문은 아직까지 보안의 연구가 미진한 XML 데이터 스트림을 자원의 제약을 갖는 클라이언트에서 처리하기 위한 방법을 제안하였다. 제안하는 기법들의 특징 첫째, 접근제어 처리 시 질의 처리와 속성 정보를 공유하여 접근제어 추가로 인한 비용 부담을 줄였다. 기존 연구는 질의 처리와 접근제어 처리를 별개의 프로세스로 다룬 반면 본 연구는 이들이 공유할 수 있는 부분(DTD 해시)을 찾아내어 추가적인 비용을 줄였다. 둘째, XPath 경로 최적화를 통해 전처리 비용을 줄이고 질의 처리 성능을 개선하였다. 즉, 입력 데이터의 정보를 계산하여 DTD 상의 정확한 위치를 찾아내고 질의 계산에 필요한 노드를 줄임으로써 질의 최적화를 가능하게 했다. 따라서 전처리 과정에서 질의와 접근제어 규칙 등록에 필요한 메모리 감소와 더불어 질의 처리 과정에서 비교해야 할 질의와 접근제어 규칙의 노드 수를 줄여 질의 처리 성능을 개선하는 효과를 얻을 수 있었고 이는 실험을 통해 확인하였다.

이러한 특징들은 가벼운 접근제어를 가능하게 하므로 자원 제약이 많은 클라이언트 환경에서 기존의 질의 처리보다 안전한 질의 처리를 할 수 있도록 해준다.

6. 참고논문

[1]. Trusted Computing Platform Alliance, (http://www.trustedcomputing.org/)

[2]. Microsoft Windows Media 9, (http://www.microsoft.com/windows/windowsmedia/default.mspx)

[3]. G. Kuper and F. Massaci and N. Rassadko, "Generalized XML security views," In Proc. of the 10th SACMAT, pp. 77-84, 2005.

[4]. Changwoo Byun, Seog Park "Two Phase Filtering for XML Access Control," In Proc. of the 3rd VLDB Workshop on Secure Data Management, pp. 115-130, 2006.

[5]. T. Grust. "Accelerating XPath Location Steps," In Proc. of the 21st Int'l ACM SIGMOD Conf. on Management of Data, pp. 109-120, 2002.

[6]. A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XPath 2.0, World Wide Web Consortium (W3C), 2007, (http://www.w3.org/TR/xpath20/)

[7]. Bouganim, L., Ngoc, F. D., and Pucheral, P. "Client-based access control management for XML documents," In Proc. of the 30th VLDB Conf., pp. 84-95, 2004.

[8]. Y. Diao, M. Franklin, "High-Performance XML Filtering: An Overview of YFilter", IEEE Data Engineering Bulletin, Vol. 26, Issue 1, pp. 41-48, 2003.