이동 컴퓨팅 환경에서 XML 데이타의 에너지 효율적인 방송

(Energy-efficient Broadcasting of XML Data in Mobile Computing Environments)

김 충 수 † 박 창 섭 †† 정 연 돈 †††
(Chung Soo Kim) (Chang-Sup Park) (Yon Dohn Chung)

요 약 본 논문에서는 무선 방송 환경에서 에너지 효율적인 질의 처리를 위한 XML 데이타 스트리밍 방법을 제안한다. 제안하는 방법은 원본 XML 문서의 데이타를 효과적으로 캡슐화하여 클라이언트에서의 스트림 접근 시간을 단축시킨다. 또한 연관된 데이타들 사이의 주소 정보를 이용하여 서로 다른 방식의 스트림 구조를 제안하고 이들에 대한 이벤트 구동 방식의 스트림 생성 방법과 단순 경로 질의에 대한 처리 알고리즘을 제시한다. 또한 실제 XML 데이타에 대한 실험을 통해 질의 처리 시 튜닝 성능을 크게 향상시킴을 보인다.

키워드 : XML, 스트리밍, 무선 데이타 방송, 에너지 효율성

Abstract In this paper, we propose a streaming method for XML data that supports energy-efficient processing of queries over the stream in mobile clients. We propose new stream organizations for XML data which have different kinds of addresses to related data in a stream. We describe event-driven stream generation algorithms for the proposed stream structures and provide search algorithms for simple XML path queries which leverage the access mechanisms incorporated in the stream. Experimental results show that our approaches can effectively improve the tuning time performance of user queries in a wireless broadcasting environment.

Key words : XML, Streaming, Wireless Data Broadcasting, Energy-Efficiency

1. 서 론

최근 무선 통신 및 정보 기술의 발달로 무선 정보 시스템(Wireless Information System) 분야에 대한 관심이 높아지고 있다[1-4]. 무선 정보 시스템이란 이동 통신 기술을 통하여 사용자들이 PDA나 휴대폰 등과 같은 휴대용 이동 단말 장비를 사용하여 지리적 제약을 받지 않고 정보 서비스를 제공받을 수 있는 시스템이다. 무선 방송에서 서버는 방송 채널을 통하여 정보를 송출하고, 클라이언트가 이 채널에 접속하여 서버에 대한 별도의 정보 요청 없이 원하는 정보를 검색한다[1,4].

· 본 연구는 "정보통신기초기술연구지원사업"의 부분적인 지원을 받아 수행되었음

† 학생회원 : 동국대학교 컴퓨터공학과
rona1st@hanmail.net
†† 정 회 원 : 수원대학교 인터넷정보공학과 교수
cspark0@gmail.com
††† 종신회원 : 고려대학교 컴퓨터학과 교수
ydchung@korea.ac.kr

논문접수 : 2005년 8월 11일
심사완료 : 2005년 10월 21일

XML은 인터넷 통해 데이타를 표현하고 교환하기 위해 사용되는 산업계 표준 마크업 언어로서, 우수한 확장성 및 유연성을 바탕으로 그룹웨어, 전자상거래, 어플리케이션 통합 등 다양한 분야에서 활용되고 있으며 그 응용 범위가 점차 확대되고 있다. 최근에는 디지털 오디오 방송(Digital Audio Broadcasting, DAB)이나 디지털 비디오 방송(Digital Video Broadcasting, DVB)과 같은 무선 방송 서비스 시스템에서도 전자 프로그램 가이드(Electronic Program Guide: EPG)나 교통 여행 정보(Traffic and Travel Information: TTI)와 같은 데이타를 표현하고 전송하기 위한 수단으로 사용되고 있다[5,6].

이동 단말은 일반적으로 한정된 에너지 자원을 갖고 있으므로 최대한 전력을 적게 소비하면서 방송 스트림 내에서 원하는 데이타를 찾을 수 있어야 한다. 본 논문에서는 XML 데이타를 방송하는 무선 정보 시스템을 고려하여, 이동 단말에서 XML 데이타를 에너지 효율적으로 검색할 수 있도록 지원하는 XML 스트리밍 방법

을 제안한다. 제안하는 방법의 특징은 다음과 같다.

- 스트림 데이타의 크기를 줄이고 이동 단말의 스트림 접근 시간을 단축시키기 위해, 텍스트 XML 문서 내의 종료 태그들을 제거하고 다양한 주소 정보들을 포함하는 스트리밍 데이타 구조를 정의한다.
- 스트림 데이타에 대해 경로식 형태로 표현되는 XML 질의를 효율적으로 처리하여 이동 단말의 튜닝 시간을 크게 향상시킬 수 있는, 단위 스트림 데이타들의 조직화 방법을 제안한다. 이 방법은 경로 질의 처리 시 질의에 무관한 스트림 데이타들에 대한 수신 및 접근을 피할 수 있도록 함으로써 단말기의 에너지 효율성을 높일 수 있다.
- 제안한 스트림 구조를 위한 이벤트 구동 방식의 스트림 생성 알고리즘과 이동 단말에서의 효율적인 질의 처리 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에 대한 배경 지식 및 관련 연구들을 살펴보고 연구 동기에 대해 기술한다. 3장에서 고려하는 시스템 환경에 대해 기술하고 질의 처리에 효과적인 스트림 데이타 구조 및 스트림 생성 방법을 제안한다. 4장에서는 3장에서 제안된 스트림 구조에 대한 질의 처리 방법에 대해 설명한다. 5장에서는 본 논문에서 제안한 방법에 대한 실험 결과를 분석하고, 마지막으로 6장에서 결론과 함께 향후 연구 방향을 제시한다.

2. 관련 연구 및 동기

이동 컴퓨팅 환경에서 데이타 방송 기법은 다음과 같은 장점들을 갖고 있다[3,4]. 첫째, 클라이언트들이 방송 채널을 통해 방송되는 데이타를 수신하여 이용함으로써 데이타 송신에 소요되는 에너지를 절약할 수 있다. 일반적으로 무선 통신에서 데이타 수신에 소요되는 에너지 비용은 송신에 드는 비용에 비해 훨씬 작으므로, 방송 기법을 통해 한정된 에너지 자원을 가진 이동 단말의 에너지 효율성을 향상시킬 수 있다. 둘째, 하나의 방송 채널을 모든 클라이언트들이 공유할 수 있으므로 한정된 무선 네트워크 대역폭을 효율적으로 활용할 수 있다. 셋째, P2P(Peer-to-Peer) 환경에서는 각각의 클라이언트와 서버 사이에 별도의 통신 채널이 필요하나, 방송 환경에서는 새로운 클라이언트 단말이 기존의 방송 채널에 접속함으로써 데이타를 수신할 수 있으므로 확장성이 우수하다.

무선 데이타 방송을 수신하는 이동 단말에서 방송 데이타 스트림에 대한 접근 성능은 크게 두 가지 기준, 즉, 접근 시간(access time)과 튜닝 시간(tuning time)으로 평가될 수 있다[3,4]. 접근 시간이란 방송 데이타의 검색을 위해 방송 수신을 시작한 시점부터 원하는 데이타를 모두 수신한 시점까지의 소요 시간을 나타내며, 튜닝 시간이란 이 접근 시간 동안 실제로 방송 데이타를 수신하는 시간을 나타낸다. 이동 단말은 자신이 필요로 하지 않는 데이타가 전송되는 동안은 에너지 소모량이 적은 상태, 즉, 휴지 상태(doze mode)로 있고, 실제로 방송 데이타를 수신하는 동안은 활성 상태(active mode)로 있게 된다. 다시 말하면, 튜닝 시간은 단말기가 접근 시간 동안 활성 상태로 있는 시간을 의미한다. 활성 상태에서 소모되는 에너지는 휴지 상태에서 소모되는 에너지의 수천 배에 이른다[7]. 따라서 이동 단말기의 에너지 제약성을 고려할 때, 데이타 검색 시 전력 사용을 최소화하기 위해 튜닝 시간을 단축하는 문제는 매우 중요한 연구 과제이다.

무선 방송 데이타에 대한 튜닝 시간을 줄이기 위해 (1, M) 색인, 분산 색인 등 스트림 데이타에 대한 색인(index) 정보를 함께 전송하는 방법들이 제안되었다[2,8]. 그러나 기존의 색인 구성 및 배치 방법들은 관계형 데이타와 같은 플랫(flat) 데이타를 대상으로 하기 때문에 트리 형태의 반구조적(semi-structured) 데이타인 XML 문서와 이에 대한 경로식 형태의 질의에 대해서는 직접적으로 적용되기 어렵다. 또, 일반적인 XML 색인 기법들은 시간적으로 단일 방향으로만 처리되어야 하는 스트림 데이타의 특성과 무선 네트워크의 제한된 대역폭 등으로 인해 무선 방송 환경에 적합하지 않다. 한편, 이러한 색인 정보의 중복 배치는 데이타의 접근 시간을 증가시키고 대역폭의 낭비를 초래할 수도 있다.

최근에는 XML 데이타의 스트리밍과 관련하여 XML 문서의 필터링이나 스트림 XML 데이타에 대한 질의 처리에 관한 연구 결과들이 제안되었다[9-11]. 이러한 연구들은 유한 오토마타(finite automata) 또는 푸시-다운 트랜스듀서(push-down transducer)들을 이용하여 XPath나 XQuery의 부-클래스(subclass)에 속하는 XML 질의들을 표현하고 효율적으로 처리한다. [12]는 관계형 데이타베이스 엔진에 통합 가능한, XML 스트림에 대한 반복형 XQuery 처리기를 제안하였다. 그러나 이러한 방법들은 모두 텍스트 형태의 일반적인 XML 데이타 스트림을 대상으로 보다 일반적이고 복잡한 형태의 XML 질의에 대한 효율적인 처리 방안에 초점을 두고 있으며, 본 논문과 같이 무선 환경에서의 튜닝 시간 단축을 위한 스트림 구조를 고려하고 있지 않다.

한편, XML 문서는 어플리케이션 사이의 상호운용성(interoperability)을 지원하기 위해 텍스트 형태를 취하고 있다. 일반적인 컴퓨팅 환경에서는 텍스트 표현에 따른 영향이 크지 않으나, 무선 정보 시스템이나 임베디드 시스템, 실시간 응용 등과 같이 통신 및 컴퓨팅 자원이 한정되거나 시간적인 제약이 있는 환경에서는

XML 데이타의 처리 부하가 어플리케이션의 성능에 큰 영향을 미칠 수 있다. 표 1은 널리 알려진 XML 데이타[13]들을 분석한 결과로서, 문서 전체에 대한 태그의 비율이 55~80% 정도로 비교적 큰 비중을 차지함을 알 수 있다. 특히 각 XML 요소의 경계를 나타내는 종료 태그는 문서의 데이타 양을 크게 증가시킨다. 만약 XML 문서 객체 모델에서 현재 노드의 다음에 오는 노드가 어떤 종류의 노드인지, 즉, 형제 노드인지, 자식 노드인지를 구분할 수 있으면, 종료 태그를 제거하여 문서의 용량을 크게 줄일 수 있다. 본 논문에서는 이러한 점을 이용하여 텍스트 XML 데이타를 전송 및 질의 처리에 효과적인 형태로 변환한 스트림 데이타 구조를 제안한다.

기존에 XML 문서에 대한 효율적인 표현 및 전송을 위해 XML 문서를 이진 데이타 형태로 부호화하는 방법에 대한 연구 결과들이 제안되었으며, 표준화를 위한 노력도 진행되고 있다[14-16]. 그러나 주로 XML의 기본적인 문서 구조를 효율적으로 표현하고 압축하는 방법에 관한 연구들로서, 클라이언트 단말에서의 효율적인 질의 처리를 위한, XML 데이타의 다양한 구조적 특징들에 대한 활용이 부족하다.

본 논문에서는 XML 데이타에 대한 별도의 색인 정보를 이용하지 않고 데이타 스트림에 대한 경로식 형태의 XML 질의를 효율적으로 처리할 수 있는 스트림 구조를 제안한다. 제안하는 스트림 구조는 XML 문서 요소들의 계층적 구조를 고려하여 XML 요소들을 구조적으로 연관된 요소들에 대한 여러 가지 링크 정보와 함께 캡슐화하여 전송하고 클라이언트에서 질의 처리 시 이를 활용함으로써 튜닝 시간 성능을 향상시킬 수 있다. 링크, 즉, 주소 정보는 관련된 데이타 항목들 사이의 방송 시점의 차이를 나타내므로 클라이언트 단말은 이 정보를 이용하여 휴지 상태를 유지하면서 방송 스트림 내의 불필요한 데이타들을 수신에서 제외할 수 있다. 또 스트림 내에 포함되는 XML 태그의 양을 줄임으로써 접근 시간을 단축시키는 효과를 얻을 수 있다.

3. 무선 방송을 위한 XML 스트리밍

3.1 시스템 환경

본 논문에서 가정하는 시스템 환경은 그림 1과 같다. XML 저장소는 스트리밍의 대상이 되는 원본 XML 문서들을 저장하는 데이타베이스 시스템이다. 스트림 생성 및 방송 서버는 XML 저장소로부터 송신할 XML 문서들을 선택하여 SAX 파서(parser)를 이용한 이벤트 처리 방식으로 XML 스트림 데이타를 생성하고 이를 무선 네트워크를 통해 방송한다. 스트림 데이타는 고정된 크기의 버킷(bucket) 단위로 분할되어 전송된다. 클라이언트, 즉, 이동 단말기에서는 방송 데이타의 일부를 검색하기 위해 XPath나 XQuery와 같은 표준 XML 질의 언어로 표현된 질의를 수행한다. 이 때, 방송되는 스트림 데이타를 모두 수신하는 것이 아니라 스트림 구조에 포함된 요소 링크 정보들을 활용하여 질의와 관련된 데이타만을 선택적으로 수신한다. 선택된 데이타들은 원본 XML 문서의 일부인 텍스트 XML 데이타로 변환된 후 사용자에게 전달된다.

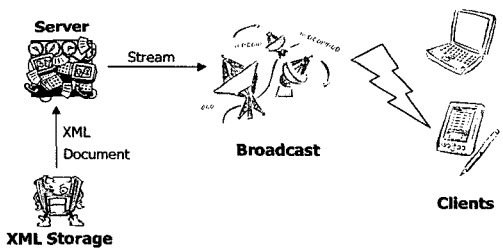

그림 1 무선 스트리밍 시스템 환경

3.2 스트림 구성

본 논문에서는 제안하는 방법은 모든 XML 요소(element)들이 시작 태그와 종료 태그를 갖고 올바르게 내포되어 있는 정형(well-formed) XML 문서들을 대상으로 한다. 이러한 정형 XML 문서는 XML 데이타 모델[17]에 제안된 바와 같이 요소 노드들로 이루어진 순서화된 트리 형태로 표현될 수 있다. 즉, 특정 요소에

표 1 XML 문서의 태그 비율

	element (bytes)	attribute (bytes)	text (bytes)	tag*/doc (%)
SigmodRecord	196594	48581	196313	55.53
e-Bay	36336	0	31386	53.65
Yahoo	24992	0	15110	62.32
Reed	265331	0	64880	80.35
UWM	1230316	114	645080	65.60
WSU	1069494	0	286093	78.89

*tag = element + attribute

내포된 요소들은 자식 노드가 되며 문서에 나타난 순서, 즉, 문서 순서(document order)에 의해 순서화된다. 이러한 일반적인 순서화된 트리는 left-child right-sibling 방식의 이진 트리 형태로 전환될 수 있다. 이진 트리의 각 내부 노드는 첫 번째 자식 노드를 가리키는 왼쪽 링크와 다음 형제 노드를 가리키는 오른쪽 링크를 가진다. 이러한 이진 트리 구조를 선-순위 깊이-우선(pre-order depth-first) 방식으로 탐색하면 원 XML 문서를 직렬화(serialization)하여 전송할 수 있다. 이 때, 자식 노드는 현재 노드의 다음에 이어서 나타나므로 특별히 고려해야 할 사항이 없으나, 형제 노드의 경우 각 노드가 각자의 자손 노드들을 가질 수 있으므로 실제로 스트림 상에서 현재 노드로부터 얼마나 떨어져 있는지를 나타내는 주소 값이 필요하다. 이 주소 값은 두 데이타 간의 방송 시간 간격을 의미한다.

본 논문에서는 XML 문서의 효율적인 스트리밍을 위해 S-노드라는 단위 노드들로 구성되는 트리 또는 그래프 구조를 제안한다. S-노드는 표 2에 기술된 바와 같이 XML 문서에 포함된 각 요소(element)에 대해 그것의 태그 이름, 연관된 속성, 텍스트 데이타 등 관련 데이타들과 스트림 상에서 미래의 시점에 전송될 다른 S-노드들에 대한 주소 정보들을 함께 캡슐화한 것이다. 각 S-노드의 처음에 저장되는 비트 플래그는 현재 S-노드가 속성, 텍스트 데이타, 특정 S-노드로의 주소 값 등을 가지고 있는지를 기술하고 스트림 상에서 다음에 출현하는 S-노드와의 관계를 명시한다. 각 플래그의 세부적인 의미는 표 3과 같다. 표 2와 표 3의 "사용" 항목은 다음 절들에서 기술되는 세 가지 스트림 구성 방식에서의 이용 여부를 나타낸다.

3.2.1 기본 구조: One Sibling Address 방법

제안하는 스트림 생성의 가장 기본적인 방법은 하나의 형제 노드 주소를 이용하는 One Sibling Address (OSA) 방법이다. 이 방식은 모든 S-노드에 스트림 상의 다음 형제 노드까지의 거리를 나타내는 주소 값을 계산하여 S-노드에 포함시킨다. 이 형제 노드 주소는 이동 클라이언트에서 방송 스트림에 대한 경로 질의 처리 시 다음 형제 노드로 효율적으로 이동하기 위해 이용될 수 있다. 그림 3은 그림 2의 예제 XML 문서를 OSA 방식의 이진 트리 구조로 표현한 것으로, 표현의 단순함을 위해 태그 이름과 링크 정보만 나타내었다.

본 논문에서는 DOM과 같이 입력 XML 문서 전체에 대한 물리적인 트리 구조를 생성하지 않고, 이벤트 구동 방식의 SAX 처리기[18]를 사용하여 S-노드 스트림을 단일 패스 스캔(scan)으로 생성한다. 즉, 입력 문서의 파싱(parsing) 과정에서 XML 요소의 시작 태그, 종료 태그, 텍스트 데이타 등이 탐지될 때마다 기정의된 관련 이벤트 처리기를 수행한다. OSA 방식의 스트림을 생성하기 위한 알고리즘은 그림 4와 같다.

특정 XML 요소(element)에 대한 S-노드 생성 시의 문제점은 현재 요소의 모든 자손 요소들에 대한 데이타

표 2 S-노드 구조

속 성	크기 (byte)	설 명	사용
flags	1	비트 플래그	모든 방법
tagName	가변	태그 이름	모든 방법
siblingAddr	1	스트림 상에서 다음 형제 노드까지의 거리(byte)	OSA
sameTagAddr	1	스트림 상에서 같은 태그 이름을 가진 다음 형제 노드까지의 거리(byte)	TSA
samePathAddr	1	스트림 상에서 같은 태그/경로 이름을 가진 다음 형제/사촌 노드까지의 거리(byte)	SPA
diffTagAddr	1	스트림 상에서 새로 나온 다른 태그 이름을 가진 다음 형제 노드까지의 거리(byte)	TSA, SPA
attributeList	가변	속성-값 리스트	모든 방법
charData	가변	텍스트 데이타	모든 방법
depth	1	문서 내의 내포된 깊이(level)	모든 방법

표 3 비트 플래그 속성

플래그 비트	설 명	사용
HAS_CHILD_NODE	스트림 상에서 다음에 오는 S-노드가 자식 노드임	모든 방법
HAS_SIBLING_NODE	스트림 상에서 다음에 오는 S-노드가 형제 노드임	모든 방법
HAS_ATTRIBUTE	속성을 가짐	모든 방법
HAS_CHAR_DATA	텍스트 데이타를 가짐	모든 방법
HAS_SIBLING_ADDR	다음 형제 노드의 주소 값을 가짐	OSA
HAS_SAME_TAG_ADDR	same-tag address 값을 가짐	TSA, SPA
HAS_DIFF_TAG_ADDR	different-tag address 값을 가짐	TSA, SPA
HAS_SAME_PATH_ADDR	same-path address 값을 가짐	SPA

```
<mondial>
    <continent id="f0_123">Asia</continent>
    <country id="f0_553" name="South Korea" capital="f0_1726">
        <name>Korea</name>
        <name>South Korea</name>
        <city id="f0_1726" country="f0_553" longitude="126.9" latitude="37.5">
            <name>Seoul</name>
            <population year="95">10229262</population>
        </city>
        <city id="f0_10230" country="f0_553">
            <name>Pusan</name>
            <population year="95">3813814</population>
        </city>
        <border length="238" country="f0_545"></border>
        <city id="f0_10235" country="f0_553">
            <name>Taegu</name>
        </city>
        <languages percentage="100">Korean</languages>
        <religions percentage="47.4">Buddhism</religions>
        <religions percentage="48.6">Christianity</religions>
    </country>
...
</mondial>
```

그림 2 예제 XML 문서

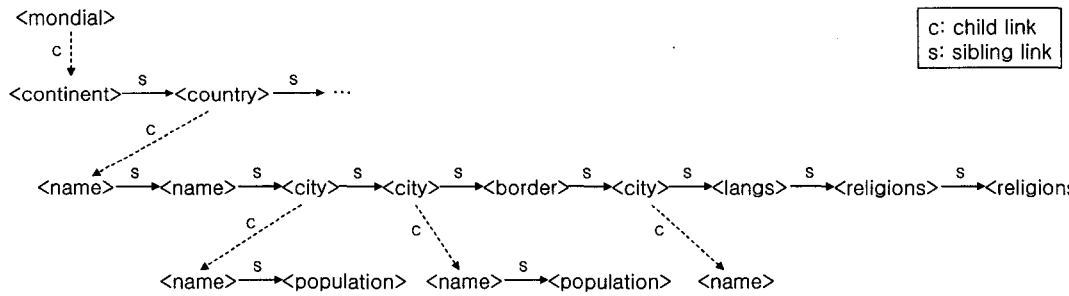

그림 3 OSA 방식으로 표현된 S-노드 트리 구조

를 입력 문서로부터 읽고 대응되는 S-노드들을 생성한 후에야 다음 형제 노드의 주소를 계산할 수 있다는 점이다. 제안하는 스트림 생성 알고리즘에서는 하나의 큐(queue)와 스택(stack)을 사용하여 이를 해결한다. 입력 문서를 스캔하면서 새로운 XML 요소의 시작 태그를 만날 때마다 형제 노드 주소 정보를 제외한 나머지 정보들로 구성된 불완전한 S-노드를 생성하여 큐에 임시 저장한다(그림 4의 33행 참조). 또 이 S-노드에 대해 스트림 상의 절대 위치와 큐 내에서의 메모리 주소 정보를 스택(stack)에 저장(push)한다(34행). 이 후 이 S-노드의 XML 요소의 다음 형제 요소의 시작 태그가 탐지되면 두 노드의 스트림 내 절대 위치로부터 두 노드 사이의 거리를 계산하여 이 S-노드의 다음 형제 노드 주소 값으로 저장한다(17~23행). 이 때, 이 S-노드에 대한 정보는 스택의 상단(top)으로부터 얻을 수 있다. 필요한 큐 공간의 크기는 루트 노드에 내포된 각 자식 노드의 서브-트리(sub-tree)의 최대 크기와 같고, 필요한 스택 공간의 크기는 XML 문서 요소의 최대 내포 깊이와 같다.

3.2.2 Two Sibling Addresses 방법

3.2.1절에서 기술한 OSA 방식의 스트림에서는 각 S-노드가 다음 형제 노드에 대한 하나의 주소만을 포함한다. 본 절에서 제안하는 Two Sibling Addresses (TSA) 방법에서는 S-노드가 서로 다른 두 종류의 형제 노드 주소를 가질 수 있다. 하나는 동일한 태그 이름을 가진 다음 형제 노드에 대한 주소(same-tag address)이고, 다른 하나는 그 S-노드 및 그 이전에 나타난 형제 노드들과 다른 태그 이름을 갖는 다음 형제 노드에 대한 주소(different-tag address)이다. 그림 5는 그림 2의 예제 XML 문서에 대한 TSA 방식의 S-노드 트리 구조를 도식화한 것이다. 그림에 나타난 바와 같이, 일단의 형제 노드들은 same-tag link와 different-tag link를 갖는 이진 트리를 구성한다. 연속된 same-tag link들은 스트림 내에서 동일한 태그 이름을 갖는 형제 S-노드들을 연결하는 체인을 형성한다.

TSA 구조의 스트림 생성을 위한 컨텐츠 핸들러는

```
1  ContentHandler StreamGenerator_OSA
2  Input: a well-formed XML data
3  Output: a stream of S-Nodes

4  // Global Variables Definitions
5  Stack S                    // stores an entry (pos, pSN)
6  Queue Q                    // stores S-Nodes
7  int curPos = 0             // current position in the stream
8  int depth = -1             // depth of the current element
9  boolean lastTag = START_TAG      // kind of the previous tag

10 EventHandler startElement (tagName, attributeList)    // invoked for a start tag
11 begin
12     depth++
13     if S is not empty then
14         if lastTag == START_TAG then    // the first child element
15             Get the top entry SE = (pos, pSN) in S.
16             Set the bit flag HAS_CHILD_NODE in pSN.
17         else    // a next sibling element
18             Pop an entry SE = (pos, pSN) from S.
19             if the bit flag HAS_CHILD_NODE in pSN is clear then
20                 Set the bit flag HAS_SIBLING_NODE in pSN.
21             end if
22             Set the bit flag HAS_SIBLING_ADDR in pSN.
23             siblingAddr of pSN = curPos - pos - (the length of the header)
24         end if
25         if depth == 1 then     // flush a root node or a sub-tree of its child
26             Flush all the S-Nodes in Q into the output stream.
27         end if
28     end if
29     Initialize an S-Node structure SN with tagName, attributeList, and depth.
30     if attributeList is not empty then
31         Set the bit flag HAS_ATTRIBUTES in SN.
32     end if
33     Enqueue SN into Q and let pSN_c be the returned address of the entry in Q.
34     Push (curPos, pSN_c) into S.
35     Increase curPos by the total length of the S-Node header and attributeList.
36     lastTag = START_TAG
37 end.

38 EventHandler endElement (tagName)   // invoked for an end tag
39 begin
40     if lastTag == END_TAG then Pop an entry from S. end if
41     Increase curPos by the length of the depth field.
42     depth--
43     lastTag = END_TAG
44 end.

45 EventHandler characters (charString)   // invoked for a character data
46 begin
47     Get the top entry SE = (pos, pSN) in S.
48     charData of pSN = charString
49     set the bit flag HAS_CHAR_DATA in pSN.
50     Increase curPos by the length of charString.
51 end.
```

그림 4 OSA 방식의 스트림 생성 알고리즘

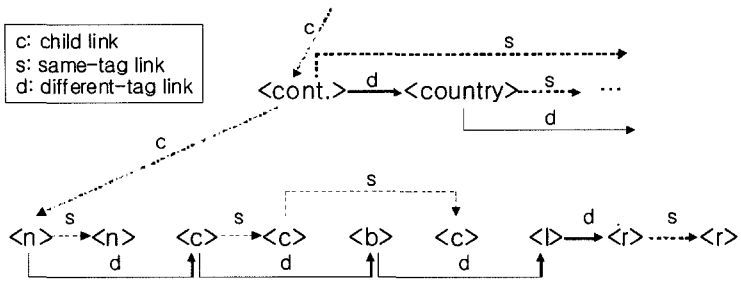

그림 5 TSA 방식의 S-노드 트리 구조

OSA 스트림 생성기로부터 확장되어 정의된다. S-노드들은 OSA 방법에서와 같이 큐와 스택을 이용하여 문서 순서대로 생성된다. 그러나 두 종류의 형제 노드 주소를 계산 및 저장하기 위해 별도의 데이타 구조들을 사용한다. 즉, same-tag address를 계산하기 위해, 현재 생성 중인 각 내포 단계의 형제 S-노드들의 집합에 대해 태그 이름을 키(key)로 하고 S-노드의 메모리 주소 및 스트림 내 절대 위치 정보를 저장하는 연관 배열(associate array)을 이용한다. 이 연관 배열은 각 태그 이름에 대해 가장 최근에 생성된(즉, same-tag chain의 끝에 해당하는) S-노드의 정보를 저장함으로써 same-tag address를 계산 및 저장하는데 이용된다. 또한 각 단계의 형제 S-노드들에 대해 새로운 태그 이름을 갖는, 가장 최근에 생성된 형제 S-노드에 대한 정보를 관리함으로써 different-tag address를 생성할 수 있다. 보다 상세한 TSA 스트림 생성 알고리즘은 [19]에 기술되어 있다.

3.2.3 Same Path Address 방법

본 절에서는 XML 스트림에 대한 보다 더 효율적인 경로 질의 처리를 위해 TSA 스트림 구조를 확장한 Same Path Address(SPA) 방식을 제안한다. SPA 방식에서는 TSA 방식에서 일단의 형제 노드들의 범위 내에서 구성한 동일 태그 노드들의 체인을 다른 서브-트리에 속하지만 같은 경로 이름을 갖는 사촌 노드를 포함하도록 연장함으로써 스트림 내에 같은 경로 이름을 갖는 S-노드들의 체인들을 구성한다. 표 2와 표 3에 나타난 바와 같이, SPA 스트림을 이루는 S-노드들은 동일한 태그 이름을 갖거나 동일한 경로 이름을 갖는 가장 가까운 다음 노드에 대한 주소를 same-path address 항목을 통해 저장하고, 비트 플래그를 이용하여 구별한다.

그림 6은 그림 2의 예제 XML 문서에 대한 SPA 방식의 S-노드 스트림 구조를 도식화한 것이다. 그림에 나타난 바와 같이, 하나의 S-노드에 대해 different-tag address를 통한 이전 형제 노드로부터의 참조 링크와 같은 경로 이름을 갖는 이전 사촌 노드로부터의 참조 링크, 즉, same-path address가 존재할 수 있으므로 SPA 스트림은 S-노드들의 트리가 아닌 유향 비순환 그래프(DAG) 구조를 갖는다.

SPA 스트림의 생성 방법은 TSA에 대한 방법과 유사하나 같은 경로 이름을 갖는 사촌 노드들에 대한 주소를 계산하기 위해 별도의 전역적인 연관 배열을 이용한다[19]. 즉, 과거에 나타난 각 경로 이름들에 대해 가장 최근에 생성된 S-노드들에 대한 정보를 관리함으로써, 동일 태그 이름을 갖는 후속 형제 노드가 존재하지 않는 노드에 대해 추후 동일 경로 이름을 갖는 사촌 노

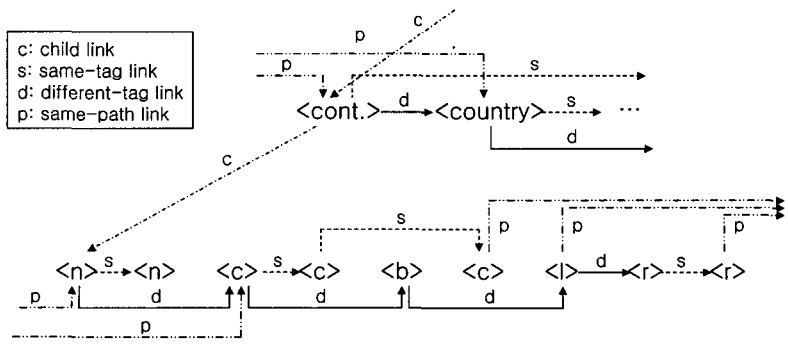

그림 6 SPA 방식의 S-노드 스트림 구조

드가 생성될 때 이에 대한 same-path address를 계산 및 저장할 수 있다.

4. 질의 처리

3장에서 제안된 방법에 의해 생성된 스트림은 전송 서버를 통해 방송되고, 이동 클라이언트는 이 스트림을 수신하여 원하는 데이타를 찾아낸다. 클라이언트가 스트림에서 데이타를 검색하는 동안 질의와 상관없는 데이타에 대해서는 이동 단말의 전원 소모를 최소화하기 위해 휴지 모드(doze mode)로 작동하는 것이 바람직하다.

본 장에서는 3장에서 제안한 세 가지 구조의 스트림 데이타에 대해 단순한 형태의 경로 질의(path query), 즉, "/a/b/c"와 같이 자식 축(child axis)과 노드 검사만을 포함하는 경로식으로 표현되는 질의를 효율적으로 처리하는 방법에 대해 기술한다. 제안하는 질의 처리 방법들은 S-노드에 포함된 서로 다른 종류의 주소 정보들을 효과적으로 이용하여 주어진 질의와 관련이 없는 데이타들을 구별해 낸다. 이러한 데이타들이 방송되는 동안 클라이언트는 에너지를 매우 적게 소비하는 휴지 모드로 전환될 수 있다.

클라이언트는 방송 스트림의 시작점부터 검색을 시작하고 경로 질의를 만족하는 S-노드에 대해 그것의 서브-트리 전체를 질의 결과로 반환한다. 반환된 S-노드들은 원 XML 문서의 문서 순서로 전송되므로 직관적인 방법으로 원래의 XML 데이타로 변환될 수 있다. 하나의 스택과 각 S-노드 안에 저장된 깊이(depth) 정보를 이용하여 결과 요소들의 종료 태그를 생성할 수 있다.

그림 7은 OSA 스트림에 대한 질의 처리 알고리즘을

```
1   Procedure Search_OSA
2   Input: a stream of S-Nodes and a simple XML path query Q
3   Output: query results R satisfying Q
4       Let S be a stack for storing S-Node headers.
5       Let pQN be a pointer to a query node, initialized with the first one in Q.
6       Let SH be an S-Node header either from the input stream or the stack S.
7       Procedure Skip_by_sibling_address (sizeOfData)
8       begin
9           sizeOfPrevData = sizeOfData
10          while SH does not have siblingAddr do
11              Pop all the entries of the previous sibling nodes of the S-Node SN from S.
12              Increase sizeOfPrevData by the total amount of data of the sibling nodes.
13              Pop an entry SH from S.  // the header of the parent node
14              Move pQN one step backward along the path of Q.
15          end while
16          Push SH into S.
17          Skip the input stream by (siblingAddr in SH - sizeOfPrevData).
18      end.
19  begin
20      while the end of stream is not detected do
21          Read the header SH of an S-Node SN from the input stream.
22          if tagName in SH equals the query node pointed by pQN then
23              if pQN points to the last query node in Q then     // an answer is found
24                  Read SN and all its descendents from the stream and insert into R.
25                  Skip_by_sibling_address (the size of the sub-tree of SN)
26              else   // query nodes remain
27                  if SN has a child S-Node as the next node then
28                      Push SH into S.
29                      Skip the data of SN.
30                      Move pQN one step forward along the path of Q.
31                  else      // SN has a sibling as the next node or is a leaf node
32                      Skip_by_sibling_address (0)
33                  end if
34              end if
35          else      // node test has failed.
36              Skip_by_sibling_address (0)
37          end if
38      end while
39  end.
```

그림 7 OSA 스트림에 대한 질의 처리 알고리즘

나타낸다. TSA와 SPA 스트림에 대한 상세한 알고리즘은 [19]에 기술되어 있다.

4.1 OSA 스트림에 대한 질의 처리

주어진 경로 질의와 일치하는 데이타를 찾기 위해서는 경로식에 포함된 질의 노드들과 일치하는 태그 이름을 갖는 S-노드들을 질의 단계의 순서대로 검색해야 한다. 검색 과정에서 현재의 질의 노드와 스트림 상의 현재 S-노드의 태그 이름이 일치하면 질의 경로식 상의 다음 질의 노드로 이동한다(그림 7의 27~30행 참조). 만일 경로식 상의 질의 노드가 남아 있으면서 현재의 S-노드가 자식 노드를 갖지 않거나(31행), 노드 검사가 실패한 경우(35행), 현재의 S-노드와 그 자손 노드들은 모두 주어진 질의의 답과 무관한 데이타이므로 현재 S-노드 내의 다음 형제 주소를 이용하여 건너 뛸 수 있다(32행 및 36행). 이 때, 만일 현재 S-노드가 다음 형제 주소 값을 갖지 않으면, 즉, 일단의 형제 노드들 중 마지막 노드일 경우, 다음 형제 노드를 갖는 가장 가까운 조상 노드를 찾아 그 형제 노드로 이동해야한다 (Skip_by_sibling_address 부-프로시져 참조). 예를 들어, 그림 3에서 마지막 "religions" 노드의 경우 부모 노드인 "country" 노드가 다음 형제 노드 주소 값을 갖고 있으므로 이 형제 노드로 이동해야 한다. 주의할 점은, 이미 검색한 이전 형제 노드들 및 그들의 자손 노드들의 데이타 크기를 고려해야만 앞으로 수신에서 제외(skip)할 실제 스트림 데이타의 양을 계산할 수 있다는 점이다. 이를 위해 스트림 검색 과정에서 각 S-노드의 비트 플래그, 태그 이름 및 형제 노드 주소 정보를 스택을 사용하여 임시 저장하고(28행) 건너뛸 스트림 데이타 양을 계산할 때 이를 이용한다(11행 및 13행).

4.2 TSA 스트림에 대한 질의 처리

3.2.2절에 기술된 바와 같이 TSA 스트림은 두 종류의 형제 노드 주소를 포함한다. 경로 질의 처리 시 이들을 활용함으로써 OSA 스트림을 이용하는 것 보다 더 많은, 질의와 무관한 데이타들을 무시할 수 있다.

TSA 질의 처리 방법에서는 일단의 형제 S-노드 집합에 대해 질의 노드와 일치하는 S-노드를 찾을 때까지는 different-tag address 링크로 연결된 일련의 노드들만을 대상으로 노드 검사를 수행하면 된다. 예를 들면, 그림 5에서 different-tag 체인에 속하지 않고 중복해서 나타나는 <n>, <c>, <r> 노드들에 대해서는 더 이상 노드 검사를 수행할 필요가 없으므로 그들의 자손 노드들을 포함하여 스트림 상의 데이타를 무시할 수 있다. 또한 질의 노드와 일치하는 S-노드가 발견된 경우 same-tag address로 연결되는 노드들을 따라감으로써 형제 노드들 중 질의 노드에 부합하는 모든 S-노드들을 선택적으로 접근할 수 있다. 그림 5에서 두 번째 <n> 노드나 첫 번째 <r> 노드와 같이 same-tag 체인이나 different-tag 체인의 마지막 노드에 대해서는 same-tag address 값을 갖는 가장 가까운 조상 노드를 찾고 실제 건너뛰어야 할 스트림 데이타의 양을 계산해야 한다. 이를 위해, OSA 질의 처리 방법에서와 같이, S-노드 검색 과정에서 스택을 사용하여 S-노드 정보를 임시 저장한다[19].

4.3 SPA 스트림에 대한 질의 처리

SPA 스트림의 S-노드들은 TSA 스트림에 포함된 두 종류의 형제 노드 주소 외에 다른 서브-트리에 속하면서 동일한 경로식을 갖는 사촌 노드에 대한 주소 정보를 갖는다. 이 부가적인 정보를 이용하면 더 효율적으로 경로 질의를 처리할 수 있다. 4.2절에 기술된 바와 같이 TSA 질의 처리 방법에서는 same-tag 체인이나 different-tag 체인의 마지막 노드에 대해서는 same-tag address 값을 갖는 가장 가까운 조상 노드를 찾기 위해 스트림 내의 트리 구조를 상향으로 탐색해야 하며 이 후 나타나는 S-노드들에 대해 질의 경로식 내의 이전 단계의 질의 노드 검사를 수행해야 한다. 그러나 SPA 스트림에서는 모든 S-노드들이 같은 경로 이름을 갖는 다음 노드에 대한 주소 정보를 갖고 있으므로 이러한 반복적인 상향 탐색을 피할 수 있다. 즉, 만일 어떤 질의 노드와 일치하는 S-노드가 same-path address를 갖지 않으면 미래의 스트림 데이타에 더 이상 그 질의 노드에 부합하는 S-노드가 존재하지 않음이 보장되므로 질의 처리를 즉시 종료할 수 있다. 따라서 OSA나 TSA 질의 처리 방법에서와 같이 루트로부터 현재 S-노드까지의 경로 상의 S-노드들의 정보를 저장하기 위한 스택이 필요하지 않고 단지 현재의 different-tag 체인 상의, 질의 노드와 부합하지 않는 형제 노드들과 그것들의 부모 노드에 대한 정보를 임시적으로 저장함으로써 자식 노드들이 모두 질의를 만족하지 않는 경우 같은 경로 이름을 갖는 다음 노드로 이동한다[19].

5. 성능 평가

본 장에서는 제안한 스트리밍 방법의 효과를 실험을 통해 평가한다. 본 실험에서는 [13]에 공개되어 있는 XML 데이타 집합들의 일부를 사용하였다. 이 데이타들은 과학, 교육, 사회, 지리, 비즈니스 등의 분야에서 생성 및 사용된 실(real) 데이타들로서, 다양한 구조와 내용을 가지고 있으며 XML에 관한 연구들에서 실험용 데이타로 널리 이용되고 있다. 이들 중 세계 지리 데이타베이스인 Mondial과 단백질 시퀀스 정보인 Swiss-Prot, 과학 논문 서지 데이타인 SIGMODRecord, NASA의 천체 정보 데이타인 Nasa 등을 본 실험에 이용하였다. 각 데이타 집합에 대한 보다 상세한 정보 및

특징은 [13]에 기술되어 있다.

이 데이타 집합들에 대해 3장에서 제안한 세 가지 구조의 S-노드 스트림을 생성하고 여러 가지 단순 경로 질의들에 대한 처리를 수행하였다. 실험에 사용된 스트림 생성 및 질의 처리 알고리즘은 Windows XP 서버 상에서 C#으로 구현하였다. 본 실험에서는 생성된 스트림이 64KB 크기의 버킷 단위로 전송된다고 가정하고, 버킷의 수를 기준으로 접근 시간과 튜닝 시간을 측정하였다.

표 4는 실험에 사용된 데이타 집합과 질의문을 나타내며, 실험 결과는 그림 8과 같다. 그림에서 "Stream"은 본 논문에서 제안한 방법으로 만들어진 스트림의 크기를, "Access"와 "Tuning"은 각각 접근 시간과 튜닝 시간을 의미한다. 결과 값들은 텍스트 XML 데이타를 원본 그대로 전송할 때의 접근 시간을 기준으로 정규화하였다. 텍스트 XML 스트림의 경우 주어진 질의 처리를 위해 스트림 전체를 탐색해야 하므로 튜닝 시간은 접근 시간과 동일하며 문서의 길이에 비례한다. 실험 결

표 4 XML 데이타 집합 및 질의문

데이타 집합	문서 크기	질의문
Mondial 3.0	3.6MB	Q1: /mondial/country/province
		Q2: /mondial/country/province/city/name
		Q3: /mondial/country/province/religions
		Q4: /mondial/river
SwissProt	112.7MB	Q5: /root/entry/org
		Q6: /root/entry/features/chain
SIGMODRecord	483.1KB	Q7: /SigmodRecord/issue/articles/article/authors/author
		Q8: /SigmodRecord/issue/articles/article/title
		Q9: /SigmodRecord/issue/volume
Nasa	24.6MB	Q10: /datasets/dataset/title
		Q11: /datasets/dataset/keywords/keyword
		Q12: /datasets/dataset/reference/source/other/author

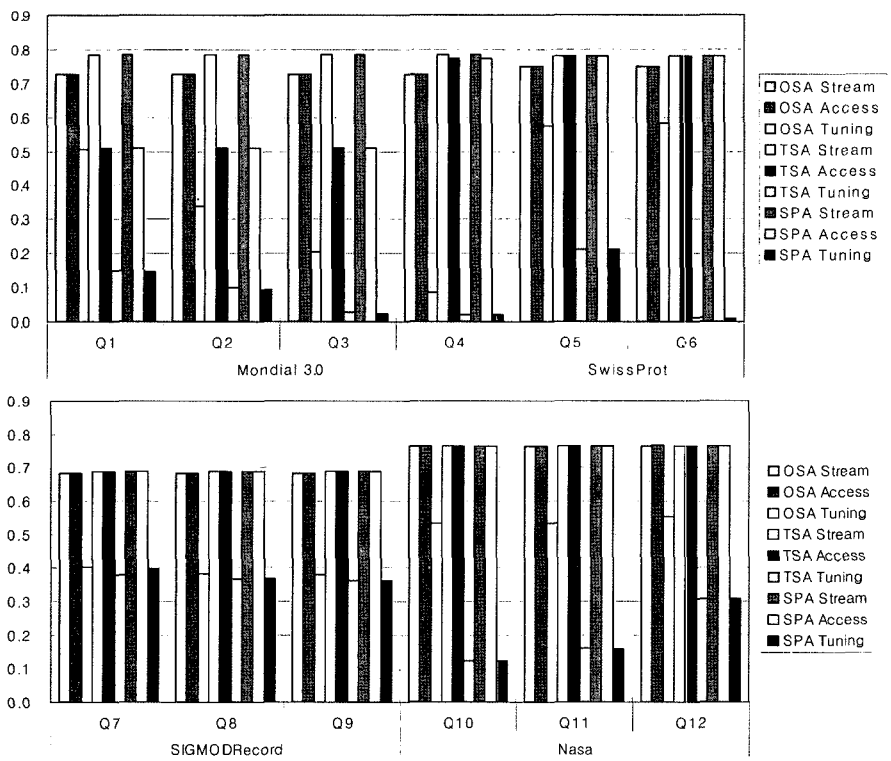

그림 8 OSA, TSA, SPA 스트림 방식에 대한 실험 결과

과에 의하면, 본 논문에서 제안된 S-노드 스트림의 길이는 텍스트 스트림에 비해 평균 25% 정도 감소하였고, 접근 시간도 그것에 비례해서 감소하였다. 이것은 주로 원본 문서에서 종료 태그들을 제거한 결과이다. 특히 Mondial 데이타의 경우 TSA 및 SPA 스트림에 대한 접근 시간이 텍스트 스트림에 비해 50% 가까이 감소하였는데, 이는 same-tag/different-tag address 및 same-path address를 이용하여 주어진 데이타 및 질의에 따라 스트림 검색 도중 질의 처리를 일찍 종료할 수 있기 때문이다. 한편, 제안된 스트리밍 방법에 의해 스트림에 대한 튜닝 시간 성능도 크게 향상됨을 알 수 있다. OSA 스트림에 대한 질의의 튜닝 시간은 텍스트 스트림의 그것에 비해 최소 40% 이상 감소하였고 TSA나 SPA 방법에서는 최소 60% 이상 감소하였다. 특히 Q3, Q4, Q6 질의들의 경우 SPA 스트림에 대한 튜닝 시간이 텍스트 스트림에 대한 튜닝 시간의 3% 이하로 나타났다.

XML 문서에 동일한 태그 이름을 가진 형제 요소들이 다수 나타나고 주어진 경로 질의의 깊이가 깊을 경우 same-tag address 및 different-tag address를 통한 질의 처리 성능이 크게 향상될 수 있고, SPA 스트림의 경우 선택도가 높고 경로가 깊은 질의에 대해 same-path address 체인을 이용하여 보다 더 효율적으로 질의를 처리할 수 있다.

6. 결론 및 향후 연구 방향

본 논문에서는 무선 방송 환경에서 XML 데이타를 효율적으로 스트리밍 할 수 있는 방법을 제안하였다. XML은 최근 급속히 그 응용 분야를 넓혀가고 있는 데이타 표현 및 전송 표준 기법으로, 무선 정보 시스템에서도 그 사용이 활발해 질 것으로 예상되므로, 효율적인 XML 스트리밍을 위한 기술의 필요성은 매우 높다고 할 수 있다.

본 논문에서는 무선 방송 환경에서 XML 문서를 효과적으로 스트리밍하기 위한 캡슐화 방안과 경로 질의 처리 시 질의에 무관한 스트림 데이타들에 대한 접근을 생략함으로써 에너지 효율성을 높일 수 있는 스트림 구조를 제안하였다. 제안한 스트림 구조들에 대한 이벤트 구동 방식의 스트림 생성 방법과 질의 처리 알고리즘을 제시하였다. 실제 XML 데이타에 대한 실험 결과 제안한 방법에 의해 생성된 스트림 데이타에 대한 질의 처리 시 튜닝 시간을 크게 향상시킬 수 있었다.

본 논문에서는 단순한 경로식으로 표현되는 질의에 대해서만 연구가 이루어졌다. 향후 *, ? 등을 포함하는 부분 부합 질의나 자손 축(//)을 포함하는 질의, XML 요소나 속성에 대한 필터링 조건과 여러 개의 경로식을 포함하는 twig pattern 질의 등 더 복잡한 XML 질의를 고려한 스트림 생성 및 질의 처리 방법에 대한 연구가 필요하다. 또 S-노드로부터 텍스트 데이타를 분리하여 튜닝 시간을 더욱 감소시킬 수 있는 방법을 연구할 계획이다.

참 고 문 헌

[1] Acharya, S., "Broadcast Disks: Data Management for Asymmetric Communication Environments," Proc. of ACM SIGMOD Conference, 1995.

[2] Chung, Y. D., Kim, M. H., "An Index Replication Scheme for Wireless Data Broadcasting," Journal of Systems and Software, 2000.

[3] Imielinski, T., Badrinath, B. R., "Data Management for Mobile Computing," SIGMOD Record, 1993.

[4] Imielinski, T., et al., "Data on Air: Organization and Access," IEEE Trans. on Knowledge and Data Engineering. 1997.

[5] Transport Protocol Experts Group, http://www.tpeg.org/

[6] DVB Document A081, Digital Video Broadcasting (DVB) Transmission System for Handheld Terminals DVB-H, 2004.

[7] Chung, Y. D., Kim, M. H., "Effective Data Placement for Wireless Broadcast," Distributed and Parallel Databases, 2001.

[8] Chen, M. S., et al., "Optimizing Index Allocation for Sequential Data Broadcasting in Wireless Mobile Computing," IEEE Trans. on Knowledge and Data Engineering. 2003.

[9] Ludascher, B., et al., "A Transducer-based XML Query Processor," Proc. of Int. Conf. on VLDB, 2002.

[10] Olteanu, D., Kiesling, T., Bry, F., "An Evaluation of Regular Path Expressions with Qualifiers against XML Streams," Proc. of Int. Conf. on Data Engineering, 2003.

[11] Peng, F., Chawathe, S.S., "XPath Queries on Streaming Data." Proc. of ACM SIGMOD Conference, 2003.

[12] Josifovski, V., et al., "Querying XML Streams," VLDB Journal, 2004.

[13] UW XML Data Repository, http://www.cs.washington.edu/research/xmldatasets

[14] Lam, W.Y., et al., "XCQ: XML Compression and Querying System," Proc. of Int. WWW Conference, 2003.

[15] Liefke, H., Suciu, D., "XMill: An Efficient Compressor for XML Data," Proc. of ACM SIGMOD Conference, 2000.

[16] The XML Binary Characterization Working Group. http://www.w3.org/XML/Binary/

[17] Cowan, J., et al., XML Information Set, W3C Recommendation, 2004.

[18] Simple API for XML, http://www.saxproject.org/
[19] Park, C. S., Kim, C. S., Chung, Y. D., "Efficient Streaming of XML Data in Wireless Broadcasting Environments," Technical Report, Ubiquitous Computing Lab., Dongguk University, 2005.

김 충 수

2004년 2월 동국대학교 컴퓨터공학과 학사. 2004년 3월~현재 동국대학교 컴퓨터공학과 석사과정. 관심분야는 XML, Mobile Databases, Mobile Computing 등

박 창 섭

1995년 2월 한국과학기술원 전산학과 학사. 1997년 2월 한국과학기술원 전산학과 석사. 2002년 2월 한국과학기술원 전자전산학과 전산학전공 박사. 2002년 3월~2005년 2월 KT 서비스개발연구소 선임 보연구원. 2005년 3월~현재 수원대학교 인터넷정보공학과 전임강사. 관심분야는 XML, Web Services, OLAP, Database Systems 등

정 연 돈

1994년 2월 고려대학교 전산과학과 학사 1996년 2월 한국과학기술원 전산학과 석사. 2000년 8월 한국과학기술원 전자전산학과 전산학전공 박사. 2000년 9월~2001년 8월 한국과학기술원 정보전자연구소 Post-Doc. 연구원. 2001년 9월~2003년 2월 한국과학기술원 전자전산학과 전산학전공 연구교수. 2003년 3월~2006년 2월 동국대학교 컴퓨터공학과 교수. 2006년 3월~현재 고려대학교 컴퓨터학과 교수. 관심분야는 XML Database, Stream Data Processing, Mobile Databases, Sensor Networks, Database Systems 등