사이트의 접속 정보 유출이 없는 네트워크 트래픽 데이타에 대한 순차 패턴 마이닝

(Privacy Preserving Sequential Patterns Mining for Network Traffic Data)

김 승 우 † 박 상 현 †† 원 정 임 †††
(Seung-Woo Kim) (Sang-Hyun Park) (Jung-Im Won)

요 약 네트워크가 급속도로 발달함에 따라, 네트워크 상에서 발생되는 트래픽 데이타를 대상으로 마이닝 기법을 적용하려는 연구가 활발히 진행되고 있다. 그러나 네트워크 트래픽 데이타를 대상으로 수행되는 마이닝 작업은 네트워크 사용자의 프라이버시를 침해할 여지가 있다는 문제점이 있다. 본 논문에서는 대용량 네트워크 트래픽 데이타를 대상으로 사이트의 프라이버시를 보호하면서 마이닝 결과의 정확성과 실용성을 보장할 수 있는 효율적인 순차 패턴 마이닝 기법을 제안한다. 제안된 기법은, N-저장소 서버 모델과 정보 유지 대체 기법을 사용함으로써, 각 사이트에 저장되어 있는 네트워크 데이타를 공개하지 않은 상태에서 순차 패턴 마이닝을 수행한다. 또한 후보 패턴의 발생 여부를 신속히 결정할 수 있는 메타 테이블을 유지하여 전체 마이닝 과정이 효율적으로 진행되도록 한다. 네트워크 상에서 발생한 실제 트래픽 데이타를 대상으로 다양한 실험을 수행한 결과 제안된 기법의 효율성과 정확성을 확인할 수 있었다.

키워드 : 데이타 마이닝, 순차 패턴, 네트워크 트래픽, 프라이버시

Abstract As the total amount of traffic data in network has been growing at an alarming rate, many researches to mine traffic data with the purpose of getting useful information are currently being performed. However, network users' privacy can be compromised during the mining process. In this paper, we propose an efficient and practical privacy preserving sequential pattern mining method on network traffic data. In order to discover frequent sequential patterns without violating privacy, our method uses the N-repository server model and the retention replacement technique. In addition, our method accelerates the overall mining process by maintaining the meta tables so as to quickly determine whether candidate patterns have ever occurred. The various experiments with real network traffic data revealed the efficiency of the proposed method.

Key words : Data mining, Sequential pattern, Network traffic, Privacy

1. 서 론

네트워크가 급속도로 발달함에 따라 네트워크에 연결된 컴퓨터와 네트워크를 통해 전송되는 데이타의 양이 증가하고 있다. 최근 대용량 네트워크 트래픽 데이타를 자동으로 수집하여 마이닝 기법을 적용함으로써 유용한 정보를 추출하려는 연구[1-5]가 활발히 진행되고 있다. 군사적, 상업적 용도 등의 다양한 응용 분야에서 서버로 전송되는 비정상적인 데이타를 파악함으로써 외부로부터의 침입이나 인터넷 웜의 동작을 탐지하려는 연구가 그 예이다.

표 1은 Ethereal[1]을 사용하여 얻은 네트워크 트래픽 데이타의 예를 보인다. 테이블의 각 행은 하나의 네트워크 트래픽 데이타를 표현하며, 트래픽이 발생한 시각과 송신 및 수신지에 대한 주소, 포트 번호 등의 정보로 구성된다. 일반적인 데이타와 비교했을 때 네트워크 트래

· 본 논문은 서울시가 시행하고 서울시립대학교 "지능형 도시 사업단(스마트-유비쿼터스-시티 사업단)"이 주관하는 "스마트시티를 위한 지능형 도시정보 컨버전스 시스템 개발"사업(10561)과 2005년 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단(KRF-2005-206-D00015)의 지원을 받았습니다.

† 학생회원 : 연세대학교 컴퓨터과학과
kimsw@cs.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
sanghyun@cs.yonsei.ac.kr
††† 정 회 원 : 한양대학교 정보통신대학 연구교수
jiwon@hanyang.ac.kr

논문접수 : 2006년 6월 7일
심사완료 : 2006년 9월 4일

1) http://www.ethereal.com/

표 1 Ethereal을 사용하여 얻은 네트워크 트래픽 데이타의 예

Timestamp	source address	source port	destination address	destination port
13:37:11.950966	180.1.1.1	36872	amazon.com	www
13:37:11.954474	amazon.com	www	180.1.1.1	36872
13:37:22.384472	180.1.1.1	36915	192.168.1.3	telnet
13:37:22.385327	192.168.1.3	telnet	180.1.1.1	36915

픽 데이타는 다음과 같은 특징을 갖는다. 첫째, 연결 가능한 모든 컴퓨터가 트래픽의 대상이 되므로 매우 많은 종류의 네트워크 트래픽 데이타가 존재한다. 둘째, 빈번한 데이타 송수신으로 인하여 각 사이트에는 매우 방대한 네트워크 트래픽 데이타가 축적된다. 셋째, 네트워크 트래픽 데이타는 많은 사이트에 분산되어 저장된다.

표 2 네트워크 트래픽 데이타로부터 발견할 수 있는 순차 패턴의 예

순차 패턴 1:	192.168.1.254로부터의 데이타 수신 → 192.168.1.254로의 데이타 송신 → 192.168.1.254로의 데이타 송신 → 192.168.1.254로의 데이타 송신
순차 패턴 2:	amazon.com으로부터의 데이타 수신 → amazon.com으로의 데이타 송신

위와 같은 특징을 갖는 대용량 네트워크 트래픽 데이타의 분석을 위해서 클러스터링[4]이나 연관 규칙 발견[5]과 같은 다양한 데이타 마이닝 기법을 사용할 수 있다. 그러나 네트워크 상에서 발생한 각 트래픽 데이타 간의 의미 있는 시간적 선후 관계를 발견하기 위해서는 순차 패턴 마이닝 기법[1-3]을 활용해야 한다. 표 2는 네트워크 트래픽 데이타로부터 발견할 수 있는 순차 패턴의 예를 보여준다. 여기서, 순차 패턴 2는 다수의 사이트에서 "amazon.com"으로부터의 데이타 수신이 발생한 후에는 "amazon.com"으로의 데이타 송신이 발생함을 나타낸다.

네트워크 트래픽 데이타에는 언제 어느 곳에 접속했다는, 네트워크 사용자의 인터넷 사용 유형을 알려주는 데이타가 포함되어 있다. 그러므로 네트워크 트래픽 데이타를 대상으로 수행되는 마이닝 작업은 네트워크 사용자의 프라이버시를 침해할 여지가 있다는 문제점이 있다. 따라서 데이타의 수집 과정에서 데이타의 출처를 은닉하거나 데이타를 변형하는 등의 추가적인 기술이 요구된다. 또한 은닉 혹은 변형된 형태로 수집, 저장된 데이타를 대상으로 마이닝 결과의 정확성을 보장할 수 있는 신뢰성 있는 마이닝 기법의 개발이 요구된다.

개인의 프라이버시를 보장하면서도 마이닝을 수행할 수 있는 방법에 대해 최근 많은 연구[6-11]가 진행되고 있다. 그러나 기존 연구의 대부분은 적은 종류의 데이타를 대상으로 하거나 소수의 사이트에 저장된 데이타를 대상으로 마이닝을 수행하는 방식이므로 네트워크 트래픽 데이타의 특성에는 적합하지 않다. 즉, 기존의 방식을 네트워크 트래픽 데이타에 그대로 적용할 경우, 마이닝 결과의 부정확성 및 비실용성 등의 문제가 발생할 수 있다.

본 논문에서는 기존 방법이 가지는 문제점을 해결할 수 있는 순차 패턴 마이닝 기법에 대해 논의한다. 즉, 대용량 네트워크 트래픽 데이타를 대상으로 사이트의 프라이버시를 보호하면서 마이닝 결과의 정확성과 실용성을 보장할 수 있는 효율적인 순차 패턴 마이닝 기법을 제안한다. 제안된 기법에서는 하나의 마이닝 서버와 같이 동작할 수 있는 N-저장소(Repository) 서버 모델을 사용함으로써 데이타의 출처를 감춘 상태에서도 빈번하게 발생한 네트워크 트래픽 데이타(즉, 빈번 항목)를 발견할 수 있도록 한다. 또한 후보 패턴의 발생 여부를 신속히 결정할 수 있는 메타 테이블을 각 사이트에 유지함으로써 전체 마이닝 과정이 효율적으로 진행되도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 기존 연구를 살펴본다. 3장에서는 본 논문에서 해결하고자 하는 문제를 정의하고, 4장에서는 이를 해결하기 위해 본 논문에서 제안하는 기법을 상세히 설명한다. 5장에서는 네트워크 상에서 발생한 실제 트래픽 데이타를 대상으로 다양한 실험을 수행함으로써 제안된 기법의 효율성 및 정확성을 검증한다. 마지막으로 6장에서는 본 논문의 공헌을 요약하고 제안된 기법의 응용 분야를 간략히 기술한다.

2. 관련 연구

순차 패턴(Sequential pattern) 마이닝은 시간의 흐름에 따라 발생한 트랜잭션의 시퀀스들로부터 빈번하게 발생한 패턴을 찾기 위한 기법으로 제안되었다[12]. Srikant 등[13]은 패턴을 구성하는 인접한 항목 간의 시간 간격 범위가 주어지고 항목 간에 종속 관계가 존재하는 일반화된 순차 패턴 문제를 정의하였으며, 문제를 해결하기 위한 GSP 알고리즘을 제안하였다. 이후 더 효율적으로 순차 패턴을 찾기 위한 많은 연구[14-16]가 수행되었다. 시퀀스 데이타베이스의 일부를 프로젝션한

서브 데이타베이스로부터 빈번한 패턴을 찾는 작은 문제로 분할하고, 그 결과를 결합하여 전체에 대한 순차 패턴을 얻고자 하는 연구[14,15]가 대표적인 예이다. 또한 Kum 등의 연구[16]는 클러스터링을 통해 유사한 시퀀스를 모으고 클러스터에 대한 대표 시퀀스를 만든 후, 대표 시퀀스를 정렬(Alignment)하여 근사(Approximation) 순차 패턴을 효율적으로 찾는다.

순차 패턴에 대한 연구는 지속적으로 수행되고 있으며, 최근에는 여러 가지 상황에 적합한 방법에 대한 연구가 수행되었다. 원하는 패턴만을 필요로 하는 환경을 위해서는 주어진 제약 조건을 만족시키는 항목들로 구성된 패턴을 찾는 연구[17,18]가 수행되었다. 데이타가 동적으로 변화하는 환경을 위해서는 업데이트된 데이타베이스에 대한 순차 패턴을 효율적으로 찾는데 기존의 결과를 이용하는 연구[19]가 수행되었다.

Lee 등[1]은 순차 패턴 마이닝 기법을 이용하여 침입당한 상태와 정상 상태의 네트워크 트래픽 데이타를 비교 분석함으로써 침입 시에만 발생하는 패턴을 추출한 후, 이를 기반으로 침입 탐지 모델을 제안하여 네트워크 트래픽 데이타에 대한 마이닝 결과의 유용성을 보였다. 이 모델은 네트워크를 사용하는 사이트를 순차 패턴 마이닝의 대상인 시퀀스로 대응시키고, 사이트에서 발생한 네트워크 트래픽 데이타를 트랜잭션을 구성하는 하나의 항목으로 대응시킨다.

침입 탐지를 위해 네트워크 트래픽 데이타를 마이닝하려는 연구[2-5]는 이후에도 지속적으로 수행되었다. 그러나 이러한 연구들은 네트워크 트래픽 데이타를 모두 한 곳에 모은 상태에서 마이닝을 수행하기 때문에 마이닝 과정에서 개개인의 인터넷 사용 유형과 같은 정보가 노출되어 개인의 프라이버시가 침해될 수 있다는 문제점을 갖는다.

Clifton 등[6]은 마이닝 과정에서 프라이버시가 침해될 수 있다는 문제를 제기하였으며, 이후 이러한 문제점을 해결하기 위한 연구[7-11]가 많이 진행되고 있다. 이러한 연구는 크게 두 가지 방식으로 분류될 수 있다.

첫 번째 방식[7-9]에서는 데이타를 수집하는 과정에서 데이타를 변형하여 프라이버시의 침해 가능성을 차단하고, 변형 후의 데이타를 변형 전의 데이타 분포를 갖도록 재구성한 후 마이닝을 수행한다. 이러한 방식의 예로서, 각 사이트는 확률 분포로부터 선택된 임의의 값을 자신이 가지고 있는 수치 데이타에 더한 후 그 결과를 서버에 전달하여 사이트의 프라이버시를 지키며, 데이타를 받은 서버가 이 확률 분포를 이용하여 전체 데이타의 실제 분포를 계산해서 의사 결정 트리를 생성하는 연구[8]가 있다. 또 다른 예로서, 정보 유지 대체 기법(Retention replacement)[7,20]을 이용하여 데이타를 변형하고 재구성하는 방법이 있다. 정보 유지 대체 기법은 0과 1로 나타낼 수 있는 이진 데이타에 사용하는 정보 보호 기법으로 개별 데이타를 숨기며 0과 1의 총 개수를 찾아낼 수 있다. 각 사이트는 p의 확률로 원래의 데이타를, 1-p의 확률로 변형된 데이타를 수집하는 측에 전달한다. 데이타를 수집하는 측에서는 각 데이타에 대해 0과 1의 발생 빈도를 집계한 후 집계된 빈도와 확률 p를 이용하여 실제 0과 1의 분포 정도를 계산한다. 그러나 이 기법들은 수치 형태의 데이타 혹은 0과 1로 양분될 수 있는 데이타의 경우에만 마이닝 알고리즘을 적용할 수 있다는 한계가 있다. 두 가지 방법을 다양한 형태의 데이타에 적용하려는 연구[9,20]가 진행되었으나, 데이타가 가질 수 있는 값의 종류가 커질수록 결과의 정확성이 감소한다는 문제점이 남아있다.

두 번째 방식[10-11]에서는 각 사이트가 마이닝 과정에 참여하여 프라이버시를 침해할 수 있는 데이타는 사이트가 직접 처리하며, 서버가 중간 결과를 집계하여 최종 결과를 얻는다. 이 방식의 대표적인 예로서, Kantarcioglu 등의 연구[10]가 있다. 이 연구에서는 상호 암호화(Commutative encryption) 방법을 사용하여 각 사이트의 데이타를 수집한 후, 안전 합산(Secure sum) 방식을 사용하여 데이타의 전체 발생 빈도를 구함으로써 연관 규칙을 발견한다. 그러나 상호 암호화 및 안전 합산을 수행하기 위해서는 전체 사이트를 연결하는 사이클을 따라 데이타를 순차적으로 전송해야 하기 때문에 사이트의 수가 매우 큰 경우에는 알고리즘 수행에 많은 시간이 소요된다는 단점이 있다.

요약을 하면, 기존의 연구들을 대용량 네트워크 트래픽 데이타에 적용할 경우 다음과 같은 문제점이 발생한다. 첫째, 데이타의 종류가 다양하므로 기존의 알고리즘을 그대로 적용하기 어려우며 데이타를 원래대로 복구하는 과정에서 정확성이 감소하여 원하는 마이닝 결과를 얻을 수 없다. 둘째, 네트워크 상에는 매우 많은 수의 사이트가 존재하므로 소수의 사이트만을 대상으로 하는 마이닝 기법은 실용성 면에서 한계를 갖는다.

3. 문제 정의

네트워크 트래픽 데이타는 Ethereal과 같은 tcp/ip 캡쳐 프로그램을 사용하여 수집한다. Ethereal로부터 얻을 수 있는 정보는 표 1에서 살펴본 바와 같이 해당 트래픽이 발생한 시점의 타임스탬프와 전송한 측과 수신한 측에 대한 주소 및 포트 번호를 포함한다. 본 논문에서는 이러한 트래픽 데이타를 대상으로 각 사이트의 데이타를 공개하지 않은 상태에서 표 2와 같은 순차 패턴을 발견하고자 한다. 이를 위하여 우선 표 1 형태의 데이타

를 표 3의 형태로 재구성한다. 표 3은 사이트에서 송수신한 네트워크 트래픽 데이타를 표현한다. 여기서, "out"은 사이트가 데이타를 송신한 경우를, "in"은 수신한 경우를 나타낸다. 표 1과 표 3을 비교해 보면 표 1의 네트워크 트래픽 데이타에는 포트 정보가 포함되어 있지만 표 3의 네트워크 트래픽 데이타에는 포트 정보가 포함되어 있지 않음을 알 수 있다. 포트 정보를 생략한 것은 트래픽 데이타의 형태를 단순화하여 발생 가능한 트래픽 데이타의 종류를 줄이기 위한 것이다. 이를 통해 빈번 패턴이 발생할 확률을 높일 수 있다.

표 3 재구성된 네트워크 트래픽 데이타의 예

Timestamp	in/out	address
13:37:11.950966	out	amazon.com
13:37:11.954474	in	amazon.com
13:37:22.384472	out	192.168.1.3
13:37:22.385327	in	192.168.1.3

이렇게 재구성된 네트워크 트래픽 데이타를 대상으로 각 트래픽 데이타 간의 의미 있는 시간적 선후 관계를 발견하기 위해서는 순차 패턴 마이닝 기법[1-3]을 적용해야 한다. 이 때 다음과 같은 두 가지 이유로 인해 인접한 두 항목간의 최대 시간 간격을 설정하는 것이 바람직하다. 첫째, 인접한 두 항목간의 최대 시간 간격을 설정하지 않으면 고려해야 하는 순차 패턴의 수가 너무 많아진다. 둘째, 인접한 두 항목간의 시간 간격이 너무 큰 경우에는 두 항목이 연관되어 있다고 보기 어렵다. 이러한 문제점들을 고려하여 본 논문에서는 인접한 두 항목간의 시간 간격이 반드시 시스템에 의해 미리 정의된 최대 시간 간격(MaxGap) 보다 작거나 같아야 한다는 제약 조건을 부여한다.

위의 내용을 정리하면 본 논문에서 해결하고자 하는 문제는 다음과 같이 정의된다: 입력으로 t개의 사이트 T_1, T_2, ..., T_t, 인접한 두 항목간의 최대 시간 간격 MaxGap, 최소 지지도 MinSup가 주어지면, 인접한 두 항목간의 시간 간격이 MaxGap 이하이면서 지지도가 MinSup 이상인 모든 순차 패턴을 발견한다. 이 때 각 사이트는 표 3의 형태로 데이타를 저장하고 있으며 데이타를 외부에 공개하지 않는다고 가정한다.

4. 제안하는 순차 패턴 마이닝 기법

이번 장에서는 3장에서 정의된 문제를 해결하기 위해 본 논문에서 제안하는 기법을 설명한다. 먼저 4.1절에서는 제안하는 기법의 전체적인 흐름을 설명한다. 다음으로 4.2절과 4.3절에서는 각각 길이가 1인 빈번 패턴, 즉 빈번 항목을 발견하는 과정과 길이가 2 이상인 빈번 패턴을 발견하는 과정을 기술한다. 마지막으로 4.4절에서는 후보 패턴의 발생 여부를 신속히 판단하기 위해 각 사이트가 유지하는 메타 테이블의 구조를 기술한다.

4.1 전체적인 흐름

본 논문에서 제안하는 기법은 그림 1에서와 같이 4단계로 구성된다. 단계 1에서는 N-저장소 서버 모델을 사용하여 빈번 항목의 집합, 즉 길이가 1인 빈번 패턴의 집합 F_1을 발견한다. 다음으로 k를 1로 설정한 후 단계 2로 진행한다. 단계 2에서는 길이가 k인 빈번 패턴의 집합 F_k를 셀프 조인하여 길이가 k+1인 후보 패턴의 집합 C_{k+1}을 구한다. 만약 C_{k+1}이 공집합이면 단계 4로 진행하고, 그렇지 않으면 단계 3으로 진행한다. 단계 3에서는 C_{k+1}에 속한 각 후보 패턴에 대하여 그 후보 패턴의 발생 여부를 문의하는 질의를 모든 사이트에 보낸 후 결과를 집계하여 지지도를 구한다. 다음으로, 지지도가 MinSup 이상인 후보 패턴만을 선택하여 집합 F_{k+1}에 포함시킨 후, k를 1만큼 증가시키고 단계 2로 되돌아간다. 단계 4에서는 F_1, F_2, ..., F_k를 출력하고 마이닝 작업을 종료한다.

4.2 N-저장소 서버 모델과 빈번 항목 발견 알고리즘

본 논문에서 제안하는 N-저장소 서버 모델은 N개의 서버 $\{S_1, S_2, ..., S_N\}$와 N개의 암호화 키와 복호화 키의 쌍 $\{(EK_1, DK_1), (EK_2, DK_2), ..., (EK_N, DK_N)\}$으로 구성된다. 각 사이트는 N개의 암호화 키를 모두 보관하고 있으며, 서버 Si는 복호화 키 DK_i만을 보관하고 있다. 빈번 항목을 발견하기 위하여 N-저장소 서버 모델은 다음과 같이 작동한다.

1) 각 사이트는 표 3 형태의 트래픽 데이타들을 해쉬(Hash) 함수를 사용하여 N개의 그룹 $(G_1, G_2, ..., G_N)$으로 분할한다.

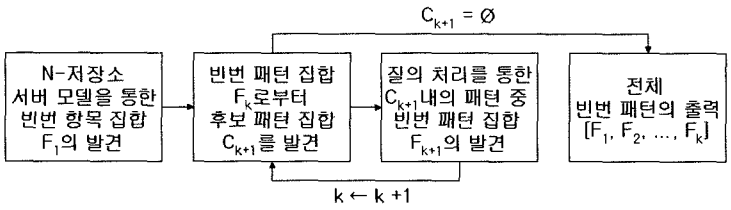

그림 1 제안하는 기법의 구성

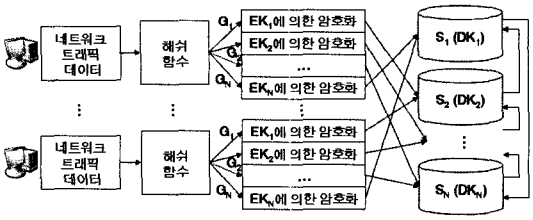

그림 2 해쉬 함수에 따른 데이타의 분할 및 암호화 과정

2) 각 사이트는 1부터 N까지의 각 i에 대하여 G_i에 속한 각각의 트래픽 데이타를 암호화 키 EK_i를 사용하여 암호화 한다.
3) 각 사이트는 1부터 N-1까지의 각 i에 대하여 G_i에 속한 각각의 암호화 된 트래픽 데이타를 서버 S_{i+1}에 전송한다. G_N에 속한 각각의 암호화 된 트래픽 데이타는 서버 S1에 전송한다. 각각의 서버 S_i는 EK_i로 암호화 된 트래픽 데이타만을 복호화 할 수 있으므로 각 사이트의 프라이버시는 보호된다.
4) 각각의 서버 S_i는 전송받은 암호화 된 데이타로부터 같은 값을 가지는 데이타의 발생 횟수를 집계하여 빈번 항목을 생성한다.
5) 암호화 된 빈번 항목을 원래 데이타로 복호화 할 수 있도록 서버로 재전송한다. 즉, 2부터 N까지의 각 i에 대하여 서버 S_i에 저장되어 있는 암호화 된 빈번 항목을 서버 S_{i-1}로 재전송한다. 서버 S_1에 저장되어 있는 암호화 된 빈번 항목은 서버 S_N으로 재전송한다.
6) 각각의 서버 S_i는 암호화 된 빈번 항목을 자신이 가지고 있는 복호화 키 DK_i를 사용하여 복호화 한다.

N-저장소 서버 모델에서 각 사이트는 동일한 해쉬 함수를 사용하여 그룹을 정하게 되며, 이에 따라 동일한 값을 가지는 데이타는 언제나 같은 그룹에 속하게 된다. 따라서 한 그룹으로부터 발견한 빈번 항목은 다른 그룹에는 속하지 않는다. 또한, 해쉬 함수는 반드시 결과를 지니므로, 존재하는 모든 빈번 항목은 반드시 어떤 그룹으로부터 발견할 수 있게 된다. 따라서 N-저장소 서버 모델을 사용하여 발견한 빈번 항목들을 모두 모은 결과는 전체 데이타에 존재하는 빈번 항목과 동일하게 된다.

4.3 길이가 2 이상인 빈번 패턴을 발견하는 알고리즘

N-저장소 서버 모델을 사용하여 빈번 항목, 즉 길이가 1인 빈번 패턴을 모두 발견한 후에는, 길이가 2 이상인 빈번 패턴을 차례로 발견해야 한다. 이를 위해 먼저 N개의 서버 중 하나를 마이닝 서버로 지정한 후, 길이가 1인 모든 빈번 패턴을 마이닝 서버로 전송한다. 마이닝 서버는 자신이 발견한 길이가 1인 빈번 패턴과 다른 서버로부터 전송받은 길이가 1인 빈번 패턴을 집합 F_1에 저장하고 k를 1로 설정한 후 다음 알고리즘을 수행한다.

1) Apriori 알고리즘[12]을 적용하여, 길이가 k인 빈번 패턴의 집합 F_k를 셀프 조인하여 길이가 k+1인 후보 패턴의 집합 C_{k+1}을 구한다. 만약 C_{k+1}이 공집합이면 단계 5로 진행하고, 그렇지 않으면 단계 2로 진행한다.
2) 마이닝 서버는 C_{k+1}에 속한 각 후보 패턴에 대하여 그 후보 패턴의 발생 여부를 문의하는 질의를 모든 사이트에 보낸다.
3) 각 사이트는 자신이 저장하고 있는 네트워크 트래픽 데이타들을 차례로 스캔하거나 4.4절에서 설명될 메타 테이블을 사용하여 후보 패턴의 발생 여부를 결정한다. 사이트의 프라이버시를 보호하기 위해서 후보 패턴의 발생 여부는 기존의 정보 유지 대체 기법[7]을 적용하여 변형한 값을 결과로 전달한다.
4) 마이닝 서버는 정보 유지 대체 기법을 적용하여 후보 패턴이 존재하는 사이트의 수를 집계한다. 다음으로, 그 값을 MinSup와 비교함으로써 각 후보 패턴의 빈번 여부를 결정한다. 최종적으로 C_{k+1}에 속한 후보 패턴 중에서 빈번하다고 판명된 것만을 선택하여 집합 F_{k+1}에 포함시킨 후, k를 1만큼 증가시키고 단계 1로 되돌아간다.
5) C_{k+1}이 공집합이라는 것은 더 이상 빈번 패턴을 발견할 수 없다는 것을 의미하므로 F_1, F_2, ..., F_k를 출력하고 마이닝 작업을 종료한다.

4.4 후보 패턴의 발생 여부를 신속하게 판단하기 위한 메타 테이블의 구조

본 절에서는 후보 패턴의 발생 여부를 신속하게 판단하기 위하여 각 사이트가 유지하는 메타 테이블의 구조를 기술한다. 이를 위해 우선 본 논문이 대상으로 하는 순차 패턴 마이닝에서는 매우 많은 수의 후보 패턴이 생성됨을 설명한다.

일반적인 Apriori 알고리즘에서는, 길이가 3인 패턴 <A, B, C>이 후보 패턴이 되기 위해서는 그의 모든 부분 패턴 (즉, <A, B>, <B, C>, <A, C>)이 빈번해야 한다. 그러나 본 논문에서 고려하는 순차 패턴 마이닝에서는 인접하지 않은 항목들로 구성된 부분 패턴 (즉, <A, C>)이 빈번하지 않더라도 인접한 항목들로 구성된 부분 패턴 (즉, <A, B>와 <B, C>)이 모두 빈번하다면 패턴 <A, B, C>는 후보 패턴이 될 수 있다. 즉, 일반적인 Apriori 알고리즘을 사용하는 다른 종류의 마이닝과 비교하여 본 논문이 대상으로 하는 순차 패턴 마이닝에서는 후보 패턴이 되기 위한 조건이 훨씬 간단하다. 이에 따라 매우 많은 수의 후보 패턴이 생성될 것이므로, 후보 패턴의 발생 여부를 각 사이트에서 얼마나 신속하게 판단할 수 있느냐가 전체 마이닝의 성능을 결

정하는 중요한 요소가 됨을 예상할 수 있다.

마이닝 서버로부터 후보 패턴의 발생 여부를 문의하는 질의가 전달되면 각 사이트는 자신이 저장하고 있는 네트워크 트래픽 데이타를 차례로 스캔하면서 후보 패턴의 발생 여부를 점검해야 한다. 최상의 상황에서는 처음 몇 개의 네트워크 트래픽 데이타만을 점검하고도 후보 패턴이 발생함을 알 수 있지만, 최악의 상황에서는 저장된 네트워크 트래픽 데이타를 모두 점검한 후에야 후보 패턴의 발생 여부를 결정할 수 있다. 본 논문에서는 후보 패턴이 발생한 경우와 그렇지 않은 경우를 모두 신속하게 판단할 수 있도록, 각 사이트 별로 최대 시간 간격 (즉, *MaxGap*) 내에서 발생한 모든 빈번 항목의 쌍과 질의된 후보 패턴 중에서 해당 사이트에서 발생한 것들을 메타 테이블 구조를 사용하여 효율적으로 저장하고 관리한다. 먼저 *MaxGap* 내에서 발생한 모든 빈번 항목의 쌍을 저장하는 메타 테이블의 구조를 살펴보자.

4.4.1 MaxGap 내에서 발생한 모든 빈번 항목의 쌍을 저장하는 메타 테이블의 구조

N-저장소 서버 모델을 사용하여 발견한 빈번 항목의 전체 수를 m이라고 하자. 우선 마이닝 서버는 빈번 항목의 목록을 각 사이트에 전달한다. 각 사이트는 빈번 항목들을 알파벳 순으로 정렬한 후 1부터 m까지의 번호를 붙여서 메타 테이블 FreqItems에 저장한다. 이 테이블은 2개의 컬럼으로 구성되며, 첫 번째 컬럼의 이름은 ItemName, 두 번째 컬럼의 이름은 Order이다.

각 사이트에서 유지하는 두 번째 메타 테이블은 OccTs_OccBits이다. 이 테이블은 세 개의 컬럼으로 구성되며, 첫 번째 컬럼의 이름은 Order, 두 번째 컬럼의 이름은 *OccTs*, 세 번째 컬럼의 이름은 *OccBits*이다. 컬럼 Order에는 빈번 항목의 순번이 저장되며, 컬럼 OccTs에는 해당 빈번 항목이 발생한 시각이 저장된다. 컬럼 *OccBits*에는 해당 빈번 항목이 발생한 시각을 기준으로 MaxGap 내에서 어떤 빈번 항목들이 발생했는가를 나타내는 정보가 비트 스트림 형태로 저장된다. 컬럼 *OccBits*는 m개의 비트로 표현되며, i번째 비트가 1이라는 것은 해당 빈번 항목이 발생한 시각을 기준으로 MaxGap 내에서 순번이 i인 빈번 항목이 발생했음을 의미한다. 표기의 편의 상 컬럼 *OccBits*의 i번째 비트를 OccBits(i)로 나타내자. 테이블 OccTs_OccBits를 구축하기 위해서는 사이트에 저장된 네트워크 트래픽 데이타들을 처음부터 끝까지 차례로 스캔해야 한다. 스캔 도중 빈번 항목이 발견되면 새로운 투플을 하나 생성하여 해당 항목의 순번과 발생 시각을 각각 컬럼 Order와 컬럼 *OccTs*에 저장한다. 또한 해당 항목의 발생 시각으로부터 MaxGap 간격 내에 발생한 모든 빈번 항목들을 조사하여 컬럼 OccBits를 구성한다. 테이블 OccTs_OccBits에 저장된 투플의 수는 사이트 내에서 각 빈번 항목이 발생한 횟수를 모두 합한 것과 같다.

각 사이트에서 유지하는 세 번째 메타 테이블은 OccCnts이다. 이 테이블은 m+1개의 컬럼으로 구성되며, 첫 번째 컬럼의 이름은 Order, 나머지 m개의 컬럼 이름은 Cnt_1, Cnt_2, ..., Cnt_m이다. 이 테이블에는 각 빈번 항목당 하나의 투플 씩, 총 m개의 투플이 저장된다. 이 테이블의 j번째 투플의 컬럼 Order에는 i가 저장되며 컬럼 Cnt_j에는 순번이 i인 빈번 항목이 발생한 후에 MaxGap 내에서 순번이 j인 빈번 항목이 발생한 횟수가 저장된다. 테이블 OccCnts의 구축을 위해 빈번 항목의 각 순번 i에 대해 다음의 SQL 문장을 수행한다.

```
insert into OccCnts(Order, Cnt1, Cnt2, ..., Cntm)
        values (
  I
  (select count(*) from OccTs_OccBits where
     Order = i and OccBits(1) = 1),
  (select count(*) from OccTs_OccBits where
     Order = i and OccBits(2) = 1),
  ...
  (select count(*) from OccTs_OccBits where
     Order = i and OccBits(m) = 1)
  );
```

그림 3은 하나의 사이트에서 구축된 메타 테이블의 예를 보인다. 테이블 FreqItems에는 세 개의 빈번 항목 A, B, C가 저장되어 있으며 각각 1, 2, 3의 순번을 가진다. 테이블 OccTs_OccBits에는 총 12개의 투플이 저장되어 있으며 각각의 투플은 컬럼 Order와 컬럼 OccTs 쌍에 의해서 구분된다. 이 테이블의 두 번째 투플을 보면 순번이 1인 항목 (즉, 항목 A)이 13:37:32.43의 시각에 발생한 후 MaxGap 내에서 순번이 2인 항목 (즉, 항목 B)만이 나타났을 뿐 다른 빈번 항목들은 나타나지 않았음을 알 수 있다. 테이블 OccCnts에는 세 개의 투플이 저장되어 있으며 각각의 투플은 컬럼 Order에 의해 구분된다. 이 테이블의 첫 번째 투플을 보면 순번이 1인 항목, 즉 항목 A가 발생한 후에 MaxGap 내에서 순번이 1인 항목 (즉, 항목 A)이 0번, 순번이 2인 항목 (즉, 항목 B)이 2번, 순번이 3인 항목 (즉, 항목 C)이 1번 발생했음을 알 수 있다.

세 개의 메타 테이블을 이용하면 네트워크 트래픽 데이타를 접근하지 않아도 후보 패턴의 발생 여부를 신속하게 판단할 수 있다. 우리는 실험을 통해 네트워크 트래픽 데이타를 직접 접근하여 후보 패턴의 발생 여부를 판단하는 방법의 성능과 세 개의 메타 테이블 만을 이용하여 후보 패턴의 발생 여부를 판단하는 방법의 성능

< FreqItems >

ItemName	Order
A	1
B	2
C	3

< OccTs_OccBits >

Order	OccTs	OccBits
1	13:37:11.95	000
1	13:37:32.43	010
1	13:38:07.05	001
1	13:38:15.51	010
2	13:37:34.21	001
2	13:38:05.44	100
2	13:38:17.23	010
2	13:38:19.12	000
3	13:37:35.17	000
3	13:38:08.54	000
3	13:38:12.31	001
3	13:38:14.08	100

< OccCnts >

Order	Cnt_1	Cnt_2	Cnt_3
1	0	2	1
2	1	1	1
3	1	0	1

그림 3 사이트에서 구성된 메타 테이블의 예

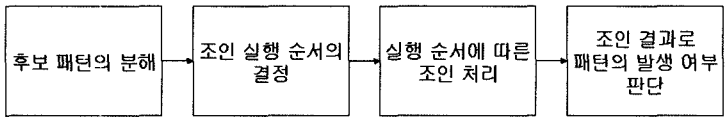

그림 4 메타 테이블을 이용하여 후보 패턴의 발생 여부를 판단하는 과정

을 비교한다.

4.4.2 메타 테이블을 이용한 질의 처리

4.4.1절에서 설명한 세 개의 메타 테이블을 이용하여 후보 패턴의 발생 여부를 판단하는 알고리즘은 그림 4와 같이 4단계로 구성된다. 단계 1에서는 후보 패턴을 다수의 부분 패턴으로 분해하며 단계 2에서는 부분 패턴의 실행 순서를 결정한다. 단계 3에서는 단계 2에서 결정된 실행 순서대로 부분 패턴들을 실행해 가면서 중간 결과들을 조인한다. 마지막으로 단계 4에서는 최종 조인 결과를 기반으로 후보 패턴의 발생 여부를 판단한다.

1) 후보 패턴의 분해

사이트에 전달된 후보 패턴의 길이가 n이라고 가정하고 이를 $CP_n = <I_1, I_2, ..., I_n>$으로 표기하자. 질의 처리의 첫 단계로서 CP_n을 길이가 2인 n-1개의 부분 패턴 $SP_j = <I_j, I_{j+1}>$ (j = 1, 2, ... n-1)으로 분해한다.

2) 부분 패턴의 실행 순서 결정

부분 패턴을 어떤 순서로 실행하고 조인하느냐에 따라 중간 결과 집합들의 크기가 달라질 수 있다. 만약 중간 결과 집합들의 크기 합을 최소화 할 수 있는 실행 순서를 결정할 수 있다면 후보 패턴의 발생 여부를 판단할 수 있는 시점을 최대한 앞당길 수 있을 것이다. 최적의 실행 순서를 결정할 수 있는 가장 간단한 방법은 가능한 모든 경우에 대하여 중간 결과 집합들의 크기 합을 구해보는 것이다. 그러나 n-1개의 부분 패턴에 대하여 총 (n-1)! 개의 서로 다른 실행 순서가 존재하므로 n이 조금만 커져도 이 방법의 성능은 급격히 떨어지게 된다. 따라서 본 논문에서는 다음과 같은 그리디(Greedy) 알고리즘을 사용하여 최적에 가까운 실행 순서를 신속하게 결정하도록 한다.

2-1) 결과 집합의 크기가 가장 작은 부분 패턴에 실행 순서 1을 할당한 후, k를 1로 초기화 한다.

2-2) 부분 패턴 SPj의 실행 순서가 k라고 하자. j'을 j-1에서 1까지 차례로 감소시키면서 실행 순서가 결정되지 않은 최초의 부분 패턴을 찾는다. 이렇게 찾은 부분 패턴을 SPj'이라고 하자. 또한 j''을 j+1에서 n-1까지 차례로 증가시키면서 실행 순서가 결정되지 않은 최초의 부분 패턴을 찾는다. 이렇게 찾은 부분 패턴을 $SP_{j''}$이라고 하자.

2-3) 만약 $SP_{j'}$과 $SP_{j''}$이 모두 존재하지 않는다면 모든 부분 패턴의 실행 순서가 결정된 것이므로 알고리즘의 수행을 종료한다. $SP_{j'}$은 존재하지만 $SP_{j''}$이 존재하지 않는 경우에는 $SP_{j'}$에 실행 순서 k+1을 할당하며, 반대로 $SP_{j''}$은 존재하지만 $SP_{j'}$이 존재하지 않는다면 $SP_{j''}$에 실행 순서 k+1을 할당한다. 만약 $SP_{j'}$과 $SP_{j''}$이 모두 존재한다면 결과 집합의 크기가 작은 것을 선택하여 실행 순서 k+1을 할당한다. $SP_{j'}$과 $SP_{j''}$의 결과 집합의 크기가 동일하다면 1에서 j'까지의 거리와 j''에서 n-1까지의 거리를 계산해서 큰 것을 선택한다. 즉, $(j'-1) \geq (n-1-j'')$ 이라면 $SP_{j'}$을, 그렇지 않다면 $SP_{j''}$을 선택한다. 이렇게 함

으로써 다음 번 단계에서 $SP_{j'}$이나 $SP_{j''}$이 존재하지 않을 확률을 줄일 수 있으며, 결과적으로 좀 더 효율적인 실행 순서를 선택할 수 있게 된다.

2-4) k를 1만큼 증가시킨 후 단계 2-2로 돌아간다.

위의 그리디 알고리즘을 수행하기 위해서는 각 부분 패턴의 결과 집합 크기를 계산하는 과정이 필요하다. 부분 패턴 SP_j = <I_j, I_{j+1}>의 결과 집합 크기란 항목 I_j가 발생한 후 MaxGap 내에서 항목 I_{j+1}이 발생한 횟수를 나타내는 것으로서 4.4.1절에서 기술한 메타 테이블 FreqItems와 OccCnts를 사용하여 다음과 같이 계산할 수 있다. 먼저 메타 테이블 FreqItems을 이용하여 항목 I_j와 I_{j+1}의 순번 p와 q를 구한다. 다음으로 메타 테이블 OccCnts를 대상으로 다음의 SQL 문을 수행하여 부분 패턴 SP_j의 결과 집합 크기를 구한다.

select Cnt_q **from** OccCnts **where** Order = p;

3) 부분 패턴의 실행 및 결과 조인

단계 2에서 결정된 순서대로 부분 패턴들을 차례로 실행해 가면서 중간 결과들을 조인한다. 즉, 1부터 n-1까지 각 k에 대하여 다음 단계들을 수행한다.

3-1) 실행 순서가 k인 부분 패턴에 대하여 이를 구성하는 두 항목의 순번을 메타 테이블 FreqItems를 이용하여 구한다. 앞 항목의 순번을 p, 뒷 항목의 순번을 q라고 하자.

3-2) 메타 테이블 OccTs_OccBits를 대상으로 다음의 SQL 문을 실행하여 실행 순서가 k인 부분 패턴의 결과 집합 RS_k를 구한다.

```
select   p, OccTs, q // p와 q는 컬럼명이
                        아니라 상수값
into     RSk
from     OccTs_OccBits
where    Order = p and OccBits(q) = 1;
```

3-3) 테이블 RS_k를 테이블 JRS_{k-1}과 조인하여 테이블 JRS_k를 생성한다. 테이블 JRS_{k-1}는 실행 순서가 1인 부분 패턴의 결과 집합부터 실행 순서가 k-1인 부분 패턴의 결과 집합까지를 모두 조인한 결과를 나타낸다. 조인 조건을 좀 더 간결하게 설명하기 위해 두 테이블을 각기 다른 이름으로 명명해보자. 즉, 후보 패턴 상에서 실행 순서가 k인 부분 패턴이 실행 순서가 1부터 k-1까지인 부분 패턴들의 왼편에 위치한다면 테이블 RS_k를 TA로, 테이블 JRS_{k-1}를 TB로 표기하자. 반대의 경우에는 JRS_{k-1}를 TA로, RS_k를 TB로 표기한다. 그러면 테이블 TA의 각 투플 ta와 테이블 TB의 각 투플 tb에 대하여 조인 조건은 다음과 같이 표현된다. 첫째, 투플 ta의 마지막 항목이 투플 tb의 첫 항목과 동일해야 한다. 둘째, 투플 tb의 첫 타임스탬프에서 투플 ta의 마지막 타임스탬프를 뺀 값이 MaxGap보다 작거나 같아야 한다.

3-4) 단계 3-3에 의해 생성된 테이블 JRS_k가 빈 테이블인가를 조사한다. 만약 빈 테이블이라면 단계 3의 실행을 중단하고 단계 4로 이동한다.

4) 후보 패턴의 발생 여부 결정

단계 3에서 생성된 최종 조인 테이블 JRS_k가 빈 테이블인가를 조사한다. 만약 빈 테이블이라면 후보 패턴이 발생하지 않았다는 결정을 내리고 빈 테이블이 아니라면 후보 패턴이 발생했다는 결정을 내린다.

예를 들어 MaxGap이 10초인 상황에서 그림 3의 메타 테이블 이용하여 후보 패턴 <A, B, C>가 사이트에서 발생했는지의 여부를 판단하는 과정을 살펴보자. 우선, 후보 패턴을 슬라이딩 방식을 이용하여 길이 2의 부분 패턴 <A, B>와 <B, C>로 분해한다. 다음으로 메타 테이블 OccCnts 테이블을 검색하여 부분 패턴 <A, B>와 <B, C>의 결과 집합의 크기를 구한 후 이를 바탕으로 실행 순서를 결정한다. 부분 패턴 <A, B>의 결과 집합의 크기는 2이고 부분 패턴 <B, C>의 결과 집합의 크기는 1이므로, <B, C>를 먼저 실행하고 <A, B>를 나중에 실행하게 된다. 부분 패턴 <B, C>와 <A, B>를 수행하면 각각 결과 집합 RS_1 = {<2, 13:37:34.21, 3>)과 RS_2 = {<1, 13:37:32.43, 2>, <1, 13:38:15.51, 2>)를 얻는다[2]. 후보 패턴 상에서 부분 패턴 <A, B>가 부분 패턴 <B, C>의 왼편에 위치하므로 RS_2를 TA로, RS_1을 TB로 치환하여 조인을 수행하면 결과로서 JRS_2 = (<1, 13:37:32.43, 2, 13:37:34.21, 3>)를 얻게 된다. 모든 부분 패턴이 실행된 상태에서 최종적인 조인 결과 JRS_2가 빈 테이블이 아니므로 후보 패턴 <A, B, C>가 사이트에서 발생했다는 결론을 내린다.

4.4.3 발생하지 않은 패턴의 신속한 검색을 위한 메타 테이블

Apriori 알고리즘에서는 길이가 n+1인 후보 패턴들을 생성하기 위해 길이가 n인 빈번 패턴의 집합을 셀프 조인한다. 이 과정에 의해 후보 패턴은 길이가 하나씩 증가하게 된다.

사이트에 전달된 길이가 n인 후보 패턴의 집합을 $\{CP_n\}$이라고 하고, $\{CP_n\}$의 원소 중에서 해당 사이트에서 발생했다고 판별된 것들의 집합을 $\{CP_n'\}$이라고 하자. 길이가 n+1인 각각의 후보 패턴 CP_{n+1}을 길이가 n인 두 개의 부분 패턴 $CP_{n+1}[1..n]$, $CP_{n+1}[2..n+1]$으로 분

2) 이 때 RS_1은 JRS_1과 동일하다.

해하면 $CP_{n+1}[1..n]$과 $CP_{n+1}[2..n+1]$이 모두 $\{CP_n\}$의 원소인 것은 당연하다. 후보 패턴 CP_{n+1}이 해당 사이트에서 발생한다는 것은 $CP_{n+1}[1..n]$과 $CP_{n+1}[2..n+1]$이 모두 사이트에서 발생한다는 것을 전제로 한다. 따라서, $CP_{n+1}[1..n] \notin \{CP_n'\}$ 이거나 $CP_{n+1}[2..n+1] \notin \{CP_n'\}$ 이라면 네트워크 트래픽 데이타나 4.4.1절에 기술된 메타 테이블을 검색하지 않고도 CP_{n+1}이 해당 사이트에서 발생하지 않음을 즉시 알 수 있다.

본 논문에서는 메타 테이블 OccCandPatt를 유지함으로써 위의 아이디어를 구현한다. 메타 테이블 OccCandPatt는 두 개의 컬럼으로 구성되며, 첫 번째 컬럼의 이름은 Len, 두 번째 컬럼의 이름은 Patt이다. 컬럼 Len에는 사이트에서 발생한 후보 패턴의 길이가 저장되며 컬럼 Patt에는 해당 후보 패턴 자체가 저장된다. 메타 테이블 OccCandPatt를 사용하여 길이가 n+1인 후보 패턴 CP_{n+1}을 처리하는 과정은 다음과 같다.

1) CP_{n+1}을 길이가 n인 두 개의 부분 패턴 $CP_{n+1}[1..n]$과 $CP_{n+1}[2..n+1]$으로 분해한다.
2) 다음 SQL 문장을 실행시켜 두 개의 테이블 TA와 TB를 구한다.
   select * **into** TA **from** OccCandPatt **where** Len = n **and** Patt = $CP_{n+1}[1..n]$;
   select * **into** TB **from** OccCandPatt **where** Len = n **and** Patt = $CP_{n+1}[2..n+1]$;
3) 두 테이블 TA와 TB 중 빈 테이블이 하나라도 있으면 후보 패턴 CP_{n+1}이 발생하지 않았다는 것을 결론을 내린다. 두 테이블이 모두 빈 테이블이 아니라면 4.4.2절에 기술된 알고리즘을 수행하여 후보 패턴 CP_{n+1}의 발생 여부를 판별한다. 만약 CP_{n+1}이 사이트에서 발생한 것으로 판별되면 CP_{n+1}의 내용을 메타 테이블 OccCandPatt에 저장한다.

위 알고리즘을 이용하면 후보 패턴이 발생하지 않음을 신속하게 판별할 수 있으나, 메타 테이블 OccCandPatt의 크기가 계속 증가하는 단점이 있다. 그러나 길이가 n+1인 후보 패턴을 점검하기 위해서는 길이가 n인 후보 패턴만이 필요하다는 것을 기억하기 바란다. 따라서, 마이닝 서버로부터 길이가 n+1인 첫 번째 후보 패턴이 도착하는 순간 메타 테이블 OccCandPatt로부터 길이가 n-1인 후보 패턴들을 모두 삭제함으로써 크기가 지속적으로 증가하는 것을 방지할 수 있다.

5. 성능 평가

본 장에서는 실험에 의한 성능 평가를 통하여 제안된 메타 테이블을 사용한 마이닝 기법의 우수성을 보인다. 5.1절에서는 실험 환경을 설명하고, 5.2절에서는 실험에 관련된 파라미터 값을 설정한다. 5.3절에서는 실험 결과를 분석한다.

5.1 실험 환경

본 실험에서는 인터넷 사용으로 인하여 발생한 실제의 네트워크 트래픽을 패킷 캡쳐 프로그램 Ethereal을 사용하여 5일간 수집하고(2005.10.5-10.9), 수집된 총 5,024,295개의 데이타로부터 인터넷 사용과 직접 관련 있는 tcp/udp 패킷에서 트래픽 발생 시각, 송수신 여부, 송수신지에 대한 주소만을 추출하여 736개의 IP 주소로 구성된 총 747,000개의 트래픽 데이타를 원본 데이타 및 질의 데이타로 사용한다. 평균 트래픽간의 발생 시각 간격은 461.38msec이다.

성능 평가는 다음 세 가지의 기법을 대상으로 한다. Naive 방식은 원본 네트워크 트래픽 데이타를 직접 액세스하여 차례로 스캔하면서 후보 패턴으로부터 빈번 패턴을 검색하는 방식으로, GSP 알고리즘[13]에 기초하고 있다. OccTs 방식은 후보 패턴을 부분 패턴들로 분해한 후, 제안된 메타 테이블 OccTs_OccBits와 OccCnts를 대상으로 검색을 수행하여 얻어진 중간 결과들을 조인함으로서 후보 패턴의 빈번 여부를 결정하는 방식이다. OccTs+OccCandPatt 방식은 후보 패턴이 발생하지 않음을 신속히 판별할 수 있는 메타 테이블 OccCandPatt를 OccTs 방식과 함께 사용한 방식이다. 또한, 이들 방식 모두 부분 패턴의 실행 순서를 결정하기 위하여 그리디 알고리즘을 채택하여 사용한다. 성능 평가 지수로 각 방식을 통하여 최대 길이가 6인 빈번 순차 패턴을 찾는데 걸린 수행 시간을 사용한다.

실험을 위한 하드웨어 플랫폼으로는 Windows XP 운영체제로 운영되고, 512MB의 메모리와 80GB (7200RPM) 디스크를 가지고 있는 Pentium IV 3.0GHz의 PC를 사용한다. 실험은 JAVA 2 Runtime Environment 1.4.2 상에서 수행한다.

5.2 실험의 기본 파라미터 값 설정

5.2.1 최소 지지도의 설정

최소 지지도 MinSup가 커질수록 빈번 패턴의 수가 급속히 감소하게 되므로 실행 시간도 이에 비례하여 감소하게 된다. 그림 5에 각각 1,000개의 트래픽을 저장하고 있는 사이트 10개에 대하여 MinSup를 변화시키면서 마이닝을 수행한 경우의 실행 시간과 빈번 패턴의 수를 보여준다. 이때 최대 시간 간격 MaxGap은 20으로 설정하였다. 실험 결과에 따르면 세 가지 방식이 모두 MinSup가 증가함에 따라 실행 시간이 감소함을 보였다. 특히 Naive 방식은 원본 데이타를 직접 액세스하여 빈번 여부를 판단하기 때문에 추출된 빈번 패턴의 수와 실행 시간이 거의 비례함을 알 수 있다. 제안된 OccTs+OccCandPatt 방식은 MinSup가 0.1인 경우 OccTs 방식 보다 실행 시간이 다소 느려지는 것을 볼 수 있는데,

이는 단지 하나의 사이트에서만 발생한 패턴까지도 모두 빈번 패턴이 되어 OccCandPatt 메타 테이블에 저장되기 때문이다. 즉, 각 사이트에서 발생한 모든 패턴이 OccCandPatt 메타 테이블에 저장되므로 그 크기가 지나치게 커지기 때문이다. 그러나, MinSup가 0.2이상에서는 제안된 OccTs+OccCandPatt 방식이 Navie 방식보다 1.47배에서 2.97배까지, OccTs 방식 보다 1.07배에서 1.09배까지 좋은 검색 성능을 보였다. 이후 실험에서는 MinSup의 기본 값을 0.2로 설정하여 실험한다.

5.2.2 최대 시간 간격의 설정

최대 시간 간격 MaxGap이 커질수록 시간 간격 내에 포함되는 패턴의 수가 증가하게 되므로 실행 시간이 증가하게 된다. 그림 6에 각각 1,000개의 트래픽을 저장하고 있는 사이트 10개에 대하여 MaxGap을 변화시키면서 마이닝을 수행한 경우의 실행 시간과 빈번 패턴의 수를 보여준다. 이때 MinSup는 0.2로 설정하였다.

실험 결과에 따르면 MaxGap이 0인 경우에는 제안된 OccTs와 OccTs+OccCandPatt 방식이 Naive 방식 보다 다소 실행 시간이 나빠짐을 보였다. 이는 MaxGap이 작은 경우에는 빈번 패턴의 수가 작기 때문에 실행 시간이 빈번 패턴을 검색하는 데 걸리는 시간 보다는 메타 테이블 OccTs를 구축하는데 걸리는 시간에 의존하기 때문이다. 그러나, MaxGap이 증가할수록 빈번 패턴의 수에 비례하여 제안된 OccTs와 OccTs+OccCandPatt 방식이 Naive 방식 보다 더 좋은 수행 시간을 보였으며, OccTs+OccCandPatt 방식이 Navie 방식 보다 1.45배에서 2.39배까지, OccTs 방식 보다 1.02배에서 1.09배까지 좋은 성능을 보였다. 이후 실험에서는 MaxGap의 기본 값을 20초로 설정하여 실험한다.

5.2.3 사이트 수와 서버 수의 설정

본 논문에서 제안하는 방식은 모두 병렬 처리 환경을 지원한다. 그림 7에 사이트 수와 서버 수의 변화에 따른 실행 시간을 측정한 결과를 보인다. 이때 각 사이트는 각각 1,000개의 트래픽을 저장하고 있으며, 사이트 수 10에서 사용된 총 10,000개의 트래픽 데이타를 사이트 수 20, 30, 40, 50에서 동일하게 사용하되, 트래픽의 수를 2배, 3배, 4배, 5배로 증가하여 사용한다. 또한, MinSup와 MaxGap은 각각 0.2와 20이다. 실험 결과에 따르면 세 가지 방식 모두 사이트 수와 서버 수의 변화에 거의 영향을 받지 않는 것으로 나타났다. 이는 동일

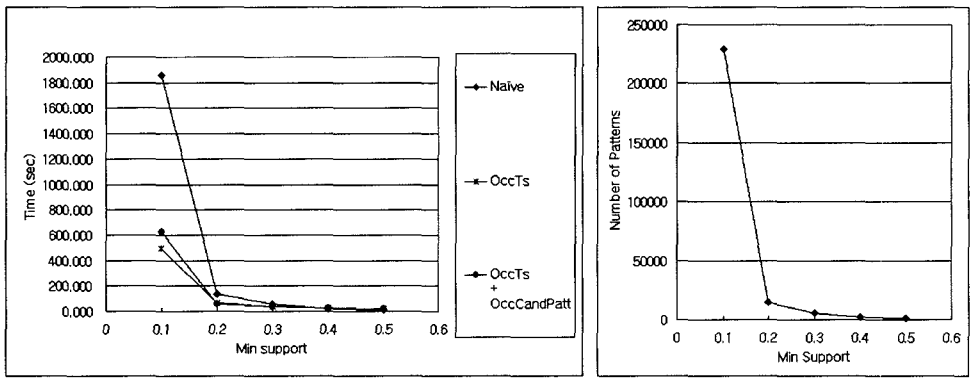

그림 5 최소 지지도 변화에 따른 실행 시간 변화와 빈번 패턴 수의 변화

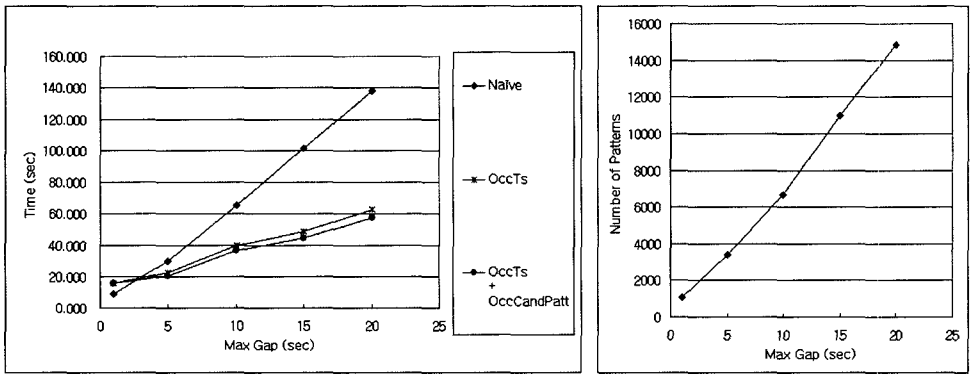

그림 6 최대 시간 간격 변화에 따른 실행 시간 변화와 빈번 패턴 수의 변화

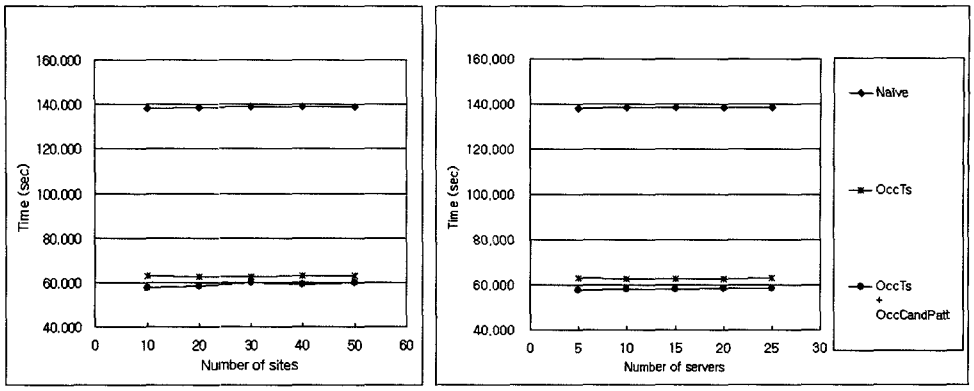

그림 7 사이트 수와 서버 수 변화에 따른 실행 시간 변화

하게 구성된 트래픽 데이타에 대하여 마이닝을 수행하므로 사이트 수와 서버 수에 무관하게 빈번 패턴의 수가 모두 동일하기 때문이다. 실험 결과에 따르면 OccTs +OccCandPatt 방식이 Navie 방식 보다 평균 2.34배, OccTs 방식 보다 평균 1.06배 좋은 성능을 보였다. 이후 실험에서는 사이트 수와 서버수를 각각 10과 5로 설정하여 실험한다.

5.3 실험 결과 및 분석

제안된 OccTs와 OccTs+OccCandPatt 방식의 성능을 평가하기 위하여 기존 방식인 Naive 방식과의 마이닝 실행 시간을 비교, 분석한다. 실험에서 MinSup와 MaxGap, 사이트 수와 서버 수는 실험 5.2절에 의하여 각각 0.2, 20, 10, 5로 설정하여 실험한다.

먼저, 각 사이트가 가지는 트래픽 수의 변화에 따른 실행 시간의 변화를 실험한다. 그림 8에 트래픽 수를 1,000부터 5,000까지 증가시키면서 실행 시간을 측정한 결과를 보인다.

실험 결과에 의하면 Naive 방식, OccTs 방식, OccTs+ OccCandPatt 방식에서 모두 트래픽 수가 증가함에 따라 실행 시간이 증가한다. 이는 트래픽 수가 증가함에 따라 빈번 패턴의 수가 증가하기 때문이다. OccTs 방식은 Naive 방식에 비해 1.60배에서 2.38배까지의 좋은 성능을 보였다. 이는 MaxGap 내에서 발생한 모든 빈번 항목의 쌍을 미리 OccTs_OccBits 메타 테이블 내에 저장하여, 이들 빈번 항목의 쌍을 조인함으로써 네트워크 트래픽 데이타를 직접 액세스하지 않아도 후보 패턴의 발생 여부를 신속하게 판단할 수 있기 때문이다. OccTs+OccCandPatt 방식은 OccTs 방식에 비해 1.01배에서 1.10배까지의 좋은 성능을 보였다. 그 이유는 메타 테이블 OccCandPatt을 이용하여 OccTs_OccBits을 검색하기 전에 후보 패턴이 해당 사이트에서 발생하지 않음을 즉시 알 수 있기 때문에, 질의 처리 시에 가지치기(pruning)효과를 얻을 수 있기 때문이다. 즉, OccTs_OccBits의 검색 대상이 되는 후보 패턴의 개수를 줄일 수 있기 때문에 전체 실행 시간을 감소시킬 수 있다. 그림 9에 OccCandPatt 메타 테이블을 이용한 가지치기의 효과를 보인다. 실험 결과에 따르면, 트래픽의 수가 증가할수록 그 효과가 급증하는 현상을 보였으며, OccTs 방식에 비해 77%에서 81%까지의 가지치기 효과가 있는 것으로 나타났다.

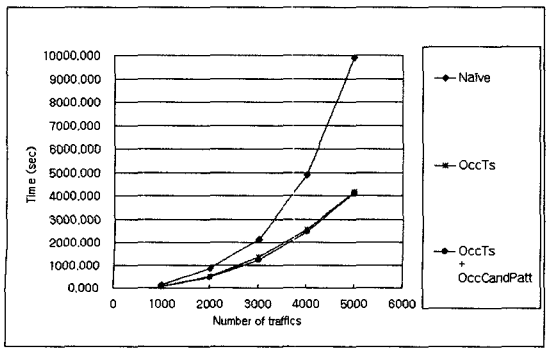

그림 8 트래픽 수 변화에 따른 실행 시간 변화

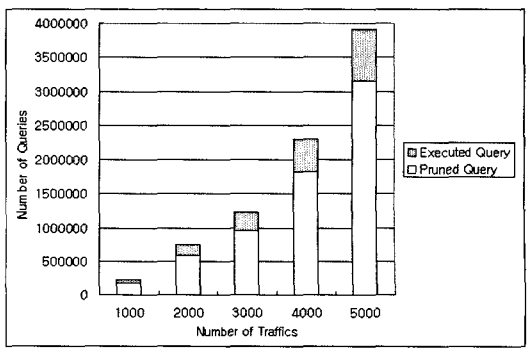

그림 9 OccCandPatt를 이용한 가지치기의 효과

표 4에 실험을 통하여 발견한 네트워크 트래픽 데이타에 대한 순차 패턴의 예를 보인다. 2번째 순차패턴은 E-mail을 받은 이후, 인쇄했다는 경우가 많다는 것을 알려주며, 4번째 순차패턴은 웹 서핑 이후, 메신저를 통해 대화를 나누는 경우가 많았다는 것을 알려준다.

표 4 실험을 통해 발견한 순차 패턴의 예

순번	순차 패턴
1	in, E-mail server → out, E-mail server
2	in, E-mail server → out, Printer server
3	in, E-mail server → out, Messenger
4	in, Web server → out, Messenger → in, Messenger
5	in, Web server → out, E-mail server → in, E-mail server → out, Web server → in, Messenger

6. 결 론

본 논문에서는 대용량 네트워크 트래픽 데이타를 대상으로 사이트의 프라이버시를 보호하면서 마이닝 결과의 정확성과 실용성을 보장할 수 있는 효율적인 순차 패턴 마이닝 기법을 제안하였다. 제안된 기법을 활용하여 침입 당한 상태와 정상 상태의 네트워크 트래픽 데이타를 분석함으로써 침입 시에만 빈번히 발생하는 순차 패턴을 추출할 수 있으며, 이를 통하여 외부로부터의 침입이나 인터넷 웜의 동작을 미리 차단할 수 있다. 또한 빈번하게 발생하는 웹 페이지의 순차적 방문 패턴을 추출하여 그 결과를 웹 페이지의 선반입(Prefetch) 및 웹 서버의 로드 밸런싱에 적용할 수 있다.

본 논문의 공헌을 요약하면 다음과 같다. 첫째, 하나의 마이닝 서버와 같이 동작할 수 있는 N-저장소 서버 모델과 후보 패턴의 발생 여부를 확률적으로 변형하여 전달하는 정보 유지 대체 기법을 사용함으로써 각 사이트에 저장되어 있는 네트워크 데이타를 공개하지 않은 상태에서도 빈번 순차 패턴을 발견할 수 있는 방법을 제시하였다. 둘째, 각 사이트에서 후보 패턴의 빈번 여부를 신속히 결정할 수 있는 메타 테이블의 구조 및 이를 이용한 질의 처리 알고리즘을 제안하였다. 셋째, 네트워크 상에서 발생한 실제 트래픽 데이타를 대상으로 다양한 실험을 수행함으로써 제안된 기법의 효율성 및 정확성을 검증하였다.

참 고 문 헌

[1] W. Lee, S. Stolfo, and K. Mok, "A Data Mining Framework for Building Intrusion Detection Models," In Proceedings of IEEE Symposium on Security and Privacy, pp. 120-132, 1999.

[2] S. Song, Z. Huang, H. Hu, and S. Jin, "A Sequential Pattern Mining Algorithm for Misuse Intrusion Detection," In Proceedings of International Workshop on Information Security and Survivability for Grid, pp. 458-465, 2004.

[3] Y. Hu and B. Panda, "A Data Mining Approach for Database Intrusion Detection," In Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 711-716, 2004.

[4] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. Tan, "Data Mining for Network Intrusion Detection," In Proceedings of NSF Workshop on Next Generation Data Mining, pp. 73-81, 2002.

[5] J. Luo and S. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," International Journal of Intelligent Systems, Vol. 15, No. 8, pp. 687-704, 2000.

[6] C. Clifton and D. Marks, "Security and Privacy Implication of Data Mining," In Proceedings of the 1996 ACM Workshop on Data Mining and Knowledge Discovery, pp. 15-19, 1996.

[7] S. Rizvi and J. Haritsa, "Maintaining Data Privacy in Association Rule Mining," In Proceedings of the 28th Conference on Very Large Data Base, pp. 682-693, 2002.

[8] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 439-450, 2000.

[9] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke "Privacy Preserving Mining of Association Rules," In Proceedings of the 2002 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217-228, 2002.

[10] M. Kantarcioglu and C. Clifton, "Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data," In Proceedings of the 2002 ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pp. 24-31, 2002.

[11] J. Zhan, L. Chang, and S. Matwin, "Privacy-Preserving Collaborative Sequential Pattern Mining," In Proceedings of Workshop on Link Analysis, Counter-terrorism and Privacy in conjunction with SIAM International Conference on Data Mining, pp. 61-72, 2004.

[12] R. Agrawal and R. Srikant, "Mining Sequential Patterns," In Proceedings of the 11th International Conference on Data Engineering, pp. 3-14, 1995.

[13] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and performance improvements," In Proceedings of the 5th International Conference on Extending Database Technology, pp. 3-17, 1996.

[14] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu, "FreeSpan: Frequent pattern-projected sequential pattern mining," In Pro-

ceedings of the 6th International Conference on Knowledge Discovery and Data Mining, pp. 355-359, 2000.
[15] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining Sequential Patterns by Pattern-growth: The PrefixSpan Approach," IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 11, pp. 1424-1440, 2004.
[16] H. Kum, J. Pei, W. Wang, and D. Duncan, "ApproxMAP: Approximate Mining of Consensus Sequential Patterns," In Proceedings of the 3rd SIAM International Conference on Data Mining, pp. 311-315, 2003.
[17] M. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," In Proceedings of 25th International Conference on Very Large Data Bases Conference, pp. 223-234, 1999.
[18] J. Pei, J. Han, and W. Wang, "Mining Sequential Patterns with Constraints in Large Databases," In Proceedings of the 11th Conference on Information and Knowledge Management, pp. 18-25, 2002.
[19] F. Masseglia, P. Poncelet, and M. Teisseire, "Incremental Mining of Sequential Patterns in Large Databases," Data and Knowledge Engineering, Vol. 46, Issue 1, pp. 97-121, 2003.
[20] R. Agrawal, R. Srikant, and D. Thomas, "Privacy Preserving OLAP," In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 251-262, 2005.

김 승 우
2005년 2월 연세대학교 컴퓨터과학과 졸업(학사). 2005년 3월~현재 연세대학교 컴퓨터과학과 석사과정. 관심분야는 데이타 마이닝, 데이타베이스 보안 등

박 상 현
정보과학회논문지 : 데이타베이스
제 33 권 제 3 호 참조

원 정 임
정보과학회논문지 : 데이타베이스
제 33 권 제 3 호 참조