論文2003-40TC-10-17

정밀 과금을 위한 컨텐츠기반 인터넷 응용 트래픽 측정 및 분석

(Content aware Internet Application Traffic Measurement and Analysis for Precise Accounting and Billing)

崔 台 相 *, 朴 貞 淑 *, 尹 承 鉉 *, 金 亨 煥 *, 金 昶 勳 *,
鄭 炯 錫 *, 李 秉 俊 *, 鄭 泰 洙 *

(Tae-Sang Choi, Jung-Sook Park, Seung-Hyun Yoon, Hyung-Hwan Kim, Chang-Hoon Kim, Hyung-Seok Cheong, Byung-Joon Lee, and Tae-Soo Jeong)

요 약

인터넷이 최선형 망에서 품질형 비즈니스 망으로 진화해가면서 정확한 트래픽 측정 데이터를 기반으로한 과금이 인터넷 서비스 제공자에게는 매우 중요한 이슈로 떠오르고 있다. 과금 협상은 서비스 제공자와 고객 간뿐만 아니라 서비스 제공자간에도 필요하지만 현재로는 가장 간단한 정액제를 대부분 사용하고 있다. 상호 관련 기관간에 동의할 수 있는 적절한 과금 정책을 찾기가 쉽지 않기도하지만 의미있는 종량제 기반의 과금 시스템을 개발하기에는 많은 기술적인 어려움이 있다. IP 헤더 정보에만 의존한 사용량 기반 측정은 피어 투 피어 응용과 네트워크 게임과 같은 인터넷 응용의 개발 및 활용으로 인한 동적인 특성으로 인해 더 이상 충분하지 않게 되었다. 이 응용들은 포트번호를 동적으로 변경하며 심지어는 여러 응용이 하나의 포트번호를 공유하기도 한다. 따라서 보다 정확하게 트래픽 사용량을 분류하고 측정할 수 있는 방법이 필요하다. 본 논문에서는 고성능, 뛰어난 적응성 및 확장성을 가지면서 정밀하게 사용량을 측정하고 분석할 수 있는 컨텐츠 기반 응용 트래픽 측정 및 분석 시스템의 구조, 알고리즘 및 구현방법을 제안한다.

Abstract

As the Internet is quickly evolving from best-effort networks to business quality networks, billing based on the precise traffic measurement becomes an important issue for Internet Service Providers (ISPs). Billing settlement is necessary not only between ISP and customers but also between ISPs. Currently, most ISPs use a flat rate charging policy. Besides the degree of difficulty in deriving appropriate charging policies agreeable by a concerned party, there are substantial technical challenges to come up with a good usage-based accounting system. Usage-based accounting depending on IP packet header information only is not sufficient anymore due to highly dynamic nature of the development and the use of the Internet applications such as peer-to-peer and network games. They are using port numbers dynamically and even several applications can use the same port number.Thus, more precise means of classifying them and accounting their traffic usages are required. In this paper, we propose a high performance, adaptable, configurable, and scalable content-aware application traffic measurement and analysis system which can achieve very accurate usage-based accounting.

Keywords : 트래픽 측정, 빌링, charging, accouting

* 正會員, 韓國電子通信研究院
(Internet Traffic Management Team, ETRI)

接受日字:2003年9月7日, 수정완료일:2003年10月10日

Ⅰ. 서 론

인터넷의 태동이후 지금까지 인터넷은 서비스 품질 및 가격면에서 모든 패킷을 동등하게 취급한 최선형 서비스를 제공해 왔다. 인터넷이 최선형 망에서 품질형 비즈니스 망으로 진화해가면서 정확한 트래픽 측정 데이터를 기반으로한 과금이 인터넷 서비스 제공자에게는 매우 중요한 이슈로 떠오르고 있다. 비록 정액제 과금 방식이 대부분의 사업자에게 적용되고 있기는 하지만 이러한 새로운 요구는 최근 많은 관심을 불러일으키고 있다. 이러한 동향의 주된 이유는 망의 성능, 서비스 품질 및 이윤을 극대화 하기 위함이다. 그러나 이 분야 연구 커뮤니티는 현재 인터넷의 특성 때문에 기술적인 해결책을 제공하기위해 심각한 도전에 직면해 있다.

주된 어려움은 현재 인터넷 응용의 개발 및 사용이 매우 동적인 특성을 보이기 때문이다. 전통적으로 인터넷은 클라이언트 서버 유형의 응용들에 주도되어 왔다. 대표적인 예로 WWW, FTP, TELNET 과 같은 것이 있다. 그러나 이러한 특성은 피어 투 피어, 멀티미디어 및 네트워크 게임과 같은 새로운 응용들이 등장함으로써 크게 바뀌게 되었다. 이러한 응용들은 포트 번호를 특정 구간내의 번호를 자유롭게 사용하거나 동적으로 할당하여 사용하기도 한다. 예를 들면 EDONKEY는 4661, 4662, 4665, 6667 등을 사용하며 RTSP 스트리밍 응용은 데이터 전송을 위해서 별도의 포트번호를 동적으로 할당한다. IANA는 응용들이 다음과 같이 포트번호를 사용하기를 권고한다. 포트번호 0 ~ 1023은 Well known 응용들이 사용하고, 1024 ~ 49151은 IANA에 등록을 한 응용이 사용하며, 49152 ~ 65535는 동적 혹은 개인적으로 임의로 사용 가능하다. 그러나 응용 개발자들은 이러한 권고를 엄격히 따르지 않고 있다. 여러 개의 인터넷 응용이 같은 포트번호를 사용하기도 하는데 일부는 방화벽을 피하기 위한 목적으로 고의로 포트번호 80번을 사용하기도 한다. 이는 포트번호와 기타 IP 패킷 헤드 정보를 기반으로한 플로우를 구분하는 방식이 더 이상 안전하거나 정확하지 않음을 의미한다. 따라서 패킷 컨텐츠의 응용 식별자를 포함한 패킷 헤드 정보가 있어야만 정확한 용용을 구별 및 측정이 가능한 것이다.

또한 인터넷은 비 대칭성을 가지기 때문에 한 응용의 트랜잭션은 일련의 요구와 응답으로 구성된 서브 트랜잭션들로 구성되며 이들은 최악의 경우 모두 다른 경로를 이용할 수 도 있다. 이러한 경우에 응용의 정확한 트래픽량 측정은 서로 다른 경로에 나타난 서브 트랜잭션 간의 상호관련성 파악과 분산 모니터링이 필요해진다. 또한 패킷의 조각화에 따른 문제도 발생한다. IP 패킷 조각들은 첫번째 조각을 제외하고 수송계층 헤더 정보를 가지지 않는다. 그러한 패킷의 경우에는 실제는 포트번호가 존재함에도 불구하고 없는 것처럼 간주된다. 조사에 따르면 인터넷백본상에서 전송되는 트래픽의 상당양이 조각화되어 있다는 것이다[1]. 이러한 추세는 IPv6, IPsec 등과 같은 터널서비스가 증가할수록 강해질 것이다. 따라서 패킷별 혹은 패킷 컨텐츠만을 기반으로 한 측정이 이 문제를 해결할 수 없으며, 더욱 정교한 플로우 기반 컨텐츠 인식 응용 사용량 측정이 필요한 것이다.

본 논문에서는 고성능, 뛰어난 적응성 및 확장성을 가지면서 정밀하게 사용량을 측정하고 분석할 수 있는 컨텐츠 기반 응용 트래픽 측정 및 분석 시스템의 구조, 알고리즘 및 구현방법을 제안한다. 본 시스템은 성능을 향상시키기 위한 전용 트래픽 측정 카드, 인터넷 응용을 컨텐츠내 식별자를 기준으로 정확하게 분류하는 에이전트, 필요시 서브 트랜잭션 플로우의 상관관계를 파악하고 에이전트에 의해서 측정된 데이터를 기반으로 트래픽 사용량을 응용의 서브 트랜잭션 수준까지 정확히 분석하는 서버로 구성된다.

본 논문에서는 먼저 관련 연구에 대해서 Ⅱ장에서 살펴본다. Ⅲ장에서는 제안된 목표를 실현하기 위한 독창적인 아이디어인 응용구분방법, 단순하지만 유연한 정책기반의 응용인식언어, 확장된 플로우 및 패킷 레코드 포멧에 대해서 기술한다. Ⅳ장에서는 이러한 아이디어들을 실현한 시스템 구조에 대해 상세하게 설명하며 Ⅴ장에서는 이 구조를 기반으로 구현된 시작품 및 현장에 설치하여 운용하면서 얻은 경험에 대해서 언급한다. 마지막으로 Ⅵ장에서 결론 및 향후 연구 방향에 대해서 정리한다.

Ⅱ. 관련 연구

지난 10여 년간 트래픽 측정 및 분석 분야에서는 많은 연구 및 개발 노력이 이루어져 왔었다. 그 결과로 많은 툴들이 소개되었는데 이들이 제공하는 기능은 다음과 같이 매우 다양한 분야에 걸쳐져 있다. 트래픽 프로

파일링, 트래픽 엔지니어링, 사이버 공격 검색, QoS 모니터링 및 사용량기반 과금정보 수집 등이 대표적인 예라 할 수 있다. CAIDA의 OCxmon[2], Tcpdump[3], Ethereal[4], 및 SPRINT의 IPMon[5] 등은 패킷의 전수검사를 통해서 트래픽 프로파일링 목적으로 비실시간으로 다양한 분석 기능을 수행한다. 즉, 이 시스템들은 패킷 기반 분석 시스템이다. 시스코의 Netflow[6], CAIDA의 CoralReef[7], Flowscan[8], 그리고 NetraMet[2]은 플로우 기반의 분석 시스템이다. 이 시스템들은 IP 패킷 헤드정보를 가공하여 플로우 레코드를 생성한다. 그리고 트래픽 프로파일링, 사용량기반 어카운팅, 트래픽엔지니어링 과 같은 다양한 분석 기능을 실시간 혹은 비실시간으로 수행한다.

또한 본고에서 제안된 구조와는 다른 목적 및 기능을 가진 응용 인식 제품도 상용화 되어있다. 시스코의 NBAR (Network Based Application Recognition)[9] 는 자사 라우터 제품의 네트워크 운영체제인 IOS (Internet Operating System) 에 내장되어 기본적인 응용 인식 기능을 제공한다. 주된 목적이 응용인식을 통해서 수락제어 및 품질 제어에 있다. 예를 들면 중요하지 않은 응용이 필요이상의 자원을 이용하고 있을 경우 이에 대해 트래픽 제어 조치를 하는 경우이다. 대부분의 보안 공격 감지 시스템(IDS) 도 식별자 메칭 기능을 하는 응용 인식 모듈을 보유하고 있다. 이러한 시스템들은 정확한 분류가 가능하지 않고 통계적 분석 결과 또한 만족할만하지 못하다는 단점들을 가지고 있다. 그러나 이중 가장 큰 문제는 이들이 적응성 및 확장성이 부족하다는 점이다. 따라서 새로운 응용이 등장하면 시스템이 수정되고 일부 혹은 전체가 재설치 되어야 하는 문제가 발생한다.

따라서 인터넷에서 응용 트래픽의 정확한 사용량 산정은 패킷 컨텐츠 검색, 플로우기반 분석, 서브 트랜잭션 플로우간의 상관관계 분석, 및 실시간 패킷 수집 성능 등이 적절하게 조화가 되어야 가능한 것이다. 기술적으로 볼 때 현재의 가용한 기술로는 매우 도전적인 연구라 할 수 있다.

Ⅲ. 방법론

본 장에서는 서론에서 소개되었던 문제점들을 해결하기위한 독창적인 방법론을 제안한다. 먼저 다양한 인터넷 응용들을 일련의 유형으로 분류할 수 있는 체계를 기술한다. 이러한 유형을 바탕으로 응용 인식 및 분석을 쉽게 처리할 수 있는 정책 기반 응용인식 언어를 설명한다. 또한 이들 응용을 표현하기 위한 플로우와 패킷 레코드의 정의를 제시한다.

1. 응용 분류

다양한 인터넷 응용 인식을 위해서 네가지 유형으로 분류하는 방식은 우리 시스템의 독창적인 아이디어이다. 이들을 네가지 유형으로 결정하기 까지는 수백개가 넘는 대표적인 인터넷 응용들을 관찰하고, 시험하고, 검사하고, 일반화하여 최종 취합하는 과정을 거쳤다. 이를 위해서 플로우 및 패킷 샘플을 네가지 유형의 망, 대규모 캠퍼스 망, 엔트프라이즈 망, 주요 인터넷 접속점, 및 ISP 에서 수집하였다. 또한 최소한 일주일 이상의 트래픽 수집이 각 네개의 망에서 이루어졌다. 아래 설명시 유형 X 방식으로 인식된 플로우를 "X 유형 플로우"로 정의한다.

한응용은 응용 레벨 뿐만 아니라 그 응용의 서브 트랜잭션 레밸까지 분류가 된다. 각 응용은 요청, 응답, 데이터 전송 등과 같이 다수개의 서브 트랜잭션으로 구성된다. 응용들을 이정도 상세하게 구분하는 목적은 트래픽 사용량을 정확히 측정하고자 함이며 알려지지 않은 응용의 양을 가능한 줄이기 위함이다. 이 방식에 따르면 응용들을 90% 이상의 정확도로 인식할 수 있다. 고정된 포트번호를 기반으로한 응용 인식을 할 경우 일반적으로 응용 인식률은 40 ~ 60 %에 이른다.

고정포토번호 유형 : Type-FP (Fixed Port based Recognition Type)

이 유형은 응용인식이 포트번호와 응용간의 기정의된 메핑 관계를 기준으로 이루어질 경우이다. 웹, FTP, SMTP, BGP 등과 같이 잘 알려진 주요 응용 서비스들과 비교적 많이 사용되는 등록구간 포트 번호 기반의 응용 서비스 들이 이 방식으로 분류될 수 있다. 그러나 서론에서 언급되었듯이 현재 인터넷 응용들의 특성상 인식률이 많이 떨어질 수 도 있다.

패킷 컨텐츠 검사 유형 : Type-PI (Payload Inspection based Recognition Type)

이 유형은 응용인식이 포트번호와 응용 PDU (Payload Data Unit)에 존재하는 응용 식별자를 모두 이용하여 이루어지는 경우이다. 두개 혹은 그 이상의 응용이 동일한 등록구간의 포트번호를 사용할 경우 특히 효과가 있다. 또한 잘알려진 응용 서비스들도 보다 정확

한 인식률이 요구될 경우 이 방식을 사용할 수 있다.

동적 포트번호 인식 유형: Type-DP (Dynamic Port based Recognition Type)

이 유형은 응용인식이 인식하고자 하는 플로우가 아닌 본 플로우를 생성시키는 정보를 전달하는 타 플로우(예, 연결신호 정보 전달 플로우)의 Payload를 검색하여 얻어진 포트번호를 기준으로 이루어지는 경우이다. Payload 검색 측면에서는 패킷 컨텐츠 검사 유형과 비슷하나 찾고자하는 식별패턴이 곧 발생할 유발 플로우의 참조 정보를 제공하는 점에서 차이가 있다. 이 유형의 대표적인 예로 수동적(passive) FTP 를 들수 있다. 대부분의 경우 동적 포트번호 할당은 둘 혹은 그 이상의 유발 플로우를 야기 시킨다. 따라서 인식하려는 특정 플로우 뿐만 아니라, 여러 개의 링크에 비교적 긴 시간 동안 걸쳐 분산될수도 있는 유발된 플로우들을 인식하기 위해 동적 포트번호 정보를 유지하고 참조할 수 있어야 한다. 그럼에도 불구하고 유발된 플로우가 잘못 인식될 경우를 대비해서 유발된 플로우가 유발시키는 플로우와 시작 및 목적지 주소 쌍이 동일한 경우에만 국한되도록 제한을 두고 있다. 그러나 일부 복잡한 응용들은 동적으로 포트번호를 할당할 뿐만 아니라 통신할 필요가 있는 서버의 주소를 바꾸어버리는 경우도 있다. 이러한 경우에 해결하기 위해서 유발시키는 플로우의 payload에서 필요한 주소를 찾아야 한다. 한편 동적으로 할당한 포트는 다음의 두 경우 중 한가지 이다. 첫번째 경우 포트는 동적 포트 할당 보고 패킷을 전송한 송신자 측의 포트인 경우이며 두번째는 받는 쪽의 경우이다. 두가지 경우 다 동적 포트와 관련 주소의 조합이 인식 과정에서 관리되고 참조가 되어야만 한다.

역방향 참조 기반 인식 유형: Type-RR (Reverse Reference based Recognition Type)

이 유형은 응용인식이 타 링크의 Type-PI 플로우를 인식하면서 획득한 참조 정보를 바탕으로 이루어지는 경우이다. 여기에서 역플로우를 정의할 필요가 있는데, <시작 주소, 목적지 주소, 시작 포트, 목적지 포트, 프로토콜>가 <a, b, x, y, p>인 어떤 플로우 X 가 있을경우 <b, a, y, x, p>라는 Y 플로우는 X의 역플로우로 정의된다. Type-RR 방식의 목적은 type-PI 플로우의 역플로우를 인식하기 위한 방식이다. 대부분의 경우 type-RR 플로우는 IP 와 TCP 헤더만 포함한 패킷들로 구성된 TCP 세션의 제어 플로우 이거나 패킷 payload에 적절한 식별 패턴을 포함하지 않는 경우이다.

2. 정책기반 응용 인식 언어

Ⅲ장 1절에서는 4가지 응용 타입의 분류방식에 대해서 설명하였다. 이러한 분류 방식을 기준으로 실제 어떻게 패킷의 컨텐츠를 수집하고 분석할 수 있는지가 또다른 중요한 이슈이다. 이를 위해서는 고성능 수집 카드, 패킷의 컨텐츠를 검색하고 필요한 플로우 정보로 바꿀 수 있는 소프트웨어 및 하드웨어 이 정보를 상이한 응용 타입으로 분류하고 타입별 트래픽 통계를 분석할 수 있는 서버 소프트웨어들이 필요해진다. 특히, 현재 인터넷 응용들의 동적인 특성을 고려할 때, 적응성과 확장성이 뛰어난 컨텐츠 필터는 필수적인 요소가 된다.

이러한 목적을 위해서 본고에서는 정책기반 응용 인식 언어인 Application Recognition Configuration Language (ARCL), 을 제안한다. 본 언어는 패킷 컨텐츠를 수집하고 분석할 수 있는 방식을 간결하게 표기할 수 있게 해 준다. 신규 응용이나 수정된 응용의 경우 인식 모듈 코드의 작성이나 배포없이 신속하게 재설정이 가능하게 본 언어가 역할을 한다. 따라서 이러한 기능이 없는 경우 신규응용이 등장한 것을 인지하였다 하더라도 측정시스템에 반영되기까지는 꽤 장기간이 소요되는 것이 일반적인데 비해 본 언어는 단순하면서도 매우 효과적으로 응용환경의 변화에 대처할 수 있다.

구체적으로 ARCL은 다음과 같은 기능을 표기할 수 있다.

- 각 플로우가 분류될수 있는 응용의 클래스
- 각 플로우의 패킷들이 분류될수 있는 응용의 서브 그룹의 클래스
- 플로우를 각 응용에 메핑하는데 필요한 방법 및 변수들
- 패킷들을 서브그룹에 메핑하는데 필요한 방법 및 변수들

ARCL 문법은 BNF (Backus Naur Form)으로 기술되어있으며 본고에서는 전체를 설명하는 대신 대부분의 기능을 보여줄 수 있는 샘플을 예시한다. 소문자는 기정의된 키워드이며, 대문자, 숫자 및 인용부호 안의 내용은 사용자들이 정의한 것이다. ARCL 문법과 의미의 기본 계위는 3 단계로 구분하였다. 최상위 계위는 응용이며 그 다음은 대표포트 (representative port - port_rep_name) 이며 최하위는 서브그룹인 decision_group 이다. 상위와 하위 계위간에는 1 대 다의 관계를 가진다. 개념을 다음의 예제를 가지고 설명한다. 비록 대부

분의 웹 서비스가 포트 80번을 사용하지만 꽤 많은 웹 전문 서비스는 80번이 아닌 번호 (예, 8080, 5001)를 사용해서 제공하고 있다. 이 모든 포트번호는 웹 응용의 범주에 속하기는 하지만 80은 HTTP로 8080은 HTTP_ALT와 같이 상이한 포트대표이름(port_rep_ name)을 가진다. 비록 포트대표이름이 포트번호와 밀접한 관계를 가지고 있기는 하지만 이들 간의 관계가 일대일 이거나 고정된 것은 아니다. 즉, 한 포트번호가 서로 다른 포트대표이름으로 여러 개의 응용에 의해서 사용되어질 수가 있다. 플로우의 패킷은 여러 개의 서브그룹으로 다시 나뉘어질 수 있다. 예를들면, HTTP 플로우는 HTTP_REQ, HTTP_REPLY, 및 HTTP_REP_ACK 패킷들로 나뉘어질 수 있다. 한 플로우의 전체를 이루는 이러한 기본 서브그룹들을 결정그룹(decision_ group)이라고 부른다. 한 상위 카테고리의 속성과 소속 하부 카테고리는 괄호로 범위를 구분한다.

본 예제에서는 3 개의 응용과 7 개의 포트대표이름으로 구성된다. 이들 포트대표이름 중 HTTP, FTP_DATA 플로우는 type-FP이며, HTTP_ATL는 양방향 모두 type-PI 이고 소스 혹은 목적지 포트번호가 8080인 것은 해당 패턴으로 검사가 되어야 한다. EDONKEY_DOWN과 FTP_CTRL 플로우는 type-PI 혹은 type-RR 으로 분류된다. 소스 포트가 21인 플로우는 src_disc_pattern으로 검사가 이루어져야만 하며 반대 역방향 플로우인 type-PI 플로우는 원래 플로우가 생성한 참조 힌트를 이용하여 인식할 수 있다. 한편 FTP_CTRL 플로우는 동적으로 할당한 포트를 사용하는 FTP_DATA_PSV 플로우를 유도할 수 있다. 동적 포트 번호를 찾을수 있는 참조 식별자와 방법은 src_ref_pattern 문장에 명기된다.

```
application WWW {
 port_rep_name HTTP port 80 protocol TCP{
  decision_group HTTP_REQ_REP_ACK {
   src_port >= 1024 dst_port == 80}
  decision_group HTTP_REP_REQ_ACK {
   src_port == 80 dst_port >= 1024}}
                port_rep_name HTTP_ALT port 8080,
8081, 9000 protocol TCP{
                    src_disc_pattern=="HTTP" in pkt 0
2 at byte 0 - 4
                    (dst_disc_pattern=="GET" in pkt 0  3
at byte 0 - 10 ||
                    dst_disc_pattern=="POST" in pkt 0  3
at byte 0 - 10 )
                    decision_group     HTTP_ALT_REQ_
REP_ACK {
                                src_port >= 1024
                                dst_port   ==   8080   ||
dst_port == 8081 || dst_port == 9000
                    }
                    decision_group     HTTP_ALT_REP_
REQ_ACK {
                                src_port   ==   8080   ||
src_port == 8081 || src_port == 9000
                                dst_port >= 1024
                    }}
}
application EDONKEY {
          port_rep_name EDONKEY_DOWN port 4662
4666,4242,4224,4660,5555 protocol TCP{
                    dst_disc_pattern=="0xe3" in pkt 2  3
at byte 0
                    decision_group   EDONKEY_DOWN_
REQ_REP_ACK {
                                src_port >= 1024
dst_port == 4662 || dst_port == 4663 ... || dst_port == 5555
                    }
                    decision_group   EDONKEY_DOWN_
REP_REQ_ACK {
                                src_port   ==   4662   ||
src_port == 4663 ... || src_port == 5555
dst_port >= 1024
                    }}
          port_rep_name EDONKEY_UDP port 4665, 4246
protocol UDP{
                    src_disc_pattern=="0xe3" in pkt any
at byte 0
                    dst_disc_pattern=="0xe3" in pkt any
at byte 0
                    decision_group EDONKEY_UDP_REQ
{
                                src_port >= 1024
dst_port == 4665 || dst_port == 4246
                    }
                    decision_group EDONKEY_UDP_REP
{
                                src_port   ==   4665   ||
src_port == 4246
dst_port >= 1024
                    }}
}
application FTP {
          port_rep_name FTP port 21 protocol TCP{
                                src_ref_pattern=="r/227
Entering                Passive                Mode
\(\d{1,3},\d{1,3},\d{1,3},\d{1,3},(\d{1,4}),(\d{1,4})\)/Ssrc_port   =
atoi($1)*1024 + atoi($2)" in pkt any at byte 0  35 induce
FTP_DOWN_P
                    decision_group   FTP_REQ_REP_ACK
{
                                src_port >= 1024 dst_port
== 21
                    }
                    decision_group   FTP_REP_REQ_ACK
```

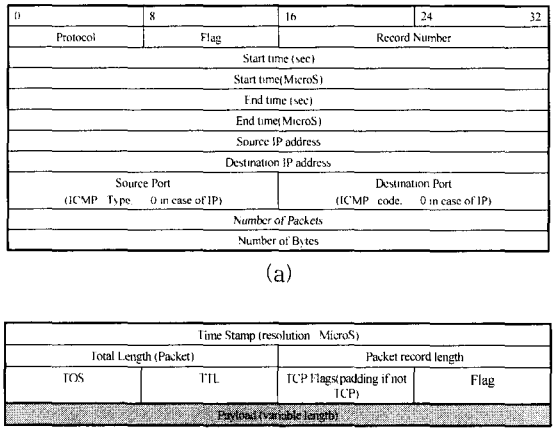
```
{
                                src_port == 21 dst_port
>= 1024
                        }}
port_rep_name FTP_DATA port 20 protocol TCP {
decision_group FTP_DATA_UP {  /* client to server */
src_port >= 1024 dst_port == 20
}
decision_group FTP_DATA_DOWN {          /* server to
client */
src_port == 20 dst_port >= 1024
}}
port_rep_name FTP_DATA_PSV port dynamic protocol TCP
{
decision_group FTP_DATA_PSV_ UP {       /* client to
server */
src_port >= 1024 dst_port == dynamic
}
decision_group FTP_DATA_PSV_DOWN {      /* server to
client */
src_port == dynamic dst_port >= 1024
}
}}

}
```

3. 컨텐츠 기반 응용 분석을 위한 플로우 정의의 확장

일단 각 패킷들이 검사되고 소속 유형으로 적절히 분류되면 이 데이터는 향후 분석 서버에서 분석을 위해 플로우 및 패킷 레코드 형태로 만들어져야 한다. 일반적으로 플로우기반 모니터링 시스템은 플로우 레코드만 생성한다. 그러나 본 시스템은 응용식별자를 포함한 payload와 패킷레코드도 생성한다. 이때 본 시스템은 모든 패킷 컨텐츠를 수집하는 것이 아니라 응용 식별자를 포함하는 패킷만을 대상으로 한다. 분석 서버는 이 정보를 각 서브 트랜잭션별 사용량 및 통합된 응용 전체의 사용량 상세 분석에 이용한다. 현재 정의된 플로우는 본 고에서 제안한 방식을 지원할 수 가 없어서 이러한 확장을 시도하였다. 이러한 레코드는 정확한 응용 인식을 위해 결정적인 역할을 한다.

<그림 1(a)>와 <그림 1(b)>는 플로우 및 패킷 레코드 포멧을 보여준다. 서버가 응용을 정확히 인식하도록 도움을 줄 정보외에 레코드 양을 가능한 얼마나 압축할 수 있느냐의 여부도 이 포멧을 설계하는데 중요한 요인이 된다. 한 플로우의 레코드는 40 바이트 고정 길이 포멧에 수용할 수 있다. 한 플로우를 결정하는 5 튜플정보는 모든 플로우 레코드에 포함된다 왜냐하면 한플로우에 속하는 모든 패킷에 공통으로 있는 정보가 5 튜플정보이기 때문이다. 플래그 비트는 액티브 타이아웃에 의해 여러 개의 서브 플로우로 나뉘어지는 장시간 플로우에 특별히 사용되는 필드이다. 또한 플로우 시작 및 종료 시간은 마이크로초로 표기되며 플로우의 패킷 및 바이트의 수는 통계처리에 사용된다. 반면 패킷 레코드는 플로우 시작시간, IP 패킷 길이, TOS (Type of Service), TTL (Time to Live), TCP 플래그 (0 는 비 TCP 패킷을 의미함) 및 payload와 같이 각 패킷의 고유한 정보로만 구성되어 있다. 비록 기본적으로 패킷 레코드의 길이가 12바이트이지만 응용식별자가 발견될 경우 이 부분의 payload가 포함된다. 이 정보는 향후 서버가 type-PI 인식 처리 및 서브그룹 분류시 사용된다. Payload가 첨부되었을겨우 PA (Payload Appended) 비트가 셋팅된다. LR (Last Record)는 플로우의 마지막 패킷임을 표시하는데 사용된다.



(a)

(b)

그림 1. (a) 플로우 레코드 (b) 패킷 레코드
Fig. 1. (a) Flow Record. (b) Packet Record.

플로우와 패킷 레코드는 파일에 저장되며 정기적으로 분석서버로 전송된다. 한 레코드 파일에는 하나의 플로우와 여기에 속한 패킷 레코드가 한 유닛으로 묶여지며 각 레코드 유닛은 플로우가 하나씩 종료시 생성된다. 서버에서 인식 처리가 파일 기준으로 이루어지므로 전송주기도 인식의 특성이나 결과에 영향을 준다. 예를들면, 전송주기가 평균 플로우 시간에 inactive timeout 시간을 더한 것 보다 짧을 경우 서로 관련있는 플로우가 나뉘어질 확률이 증가된다. 즉, 플로우를 유발시키는 플로우와 유발되는 플로우가 서로 다른 파일에 저장될 수도 있다. 긴 전송주기가 이러한 문제를 줄여주기는 하지만 본고에서는 한 파일을 처리하면서 남겨둔 참조정보로 다음 파일 처리시 활용함으로써 이 문제를 해결한다.

전송주기는 현재 분단위로 조정이 가능하며 기본값은 한 시간이다. 이정보의 주된 목적은 과금에 있으며, 따라서 준실시간 전송으로 충분하다.

Ⅳ. 시스템 구조

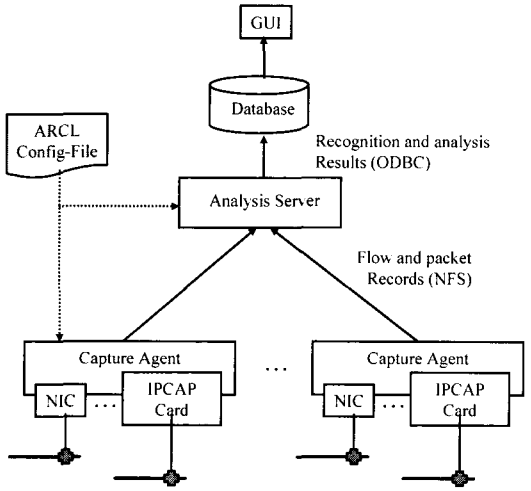


그림 2. 제안된 시스템 전체 구조도
Fig. 2. Proposed System Overall Architecture.

<그림 2>는 제안된 시스템 전체 구조를 보여준다. 수집 에이전트는 패킷을 회선으로부터 태핑하여 에이전트로 전송한다. 이때 태핑은 전기적 혹은 광신호를 직접 분리할 수 있다. 서비스 중단이 허용되지 않을 경우에는 포트 미러링을 통하여 수집할 수도 있다. 이렇게 원시 패킷을 받으면 에이전트는 먼저 비 IP 패킷을 걸러낸다. 그 다음 에이전트는 이 일련의 패킷들로부터 플로우를 추출하여 이를 간결하게 표현할 플로우 및 패킷 레코드를 생성한다. TCP, UDP 및 ICMP 패킷들은 플로우 생성과정에서 정상적으로 처리되지만 트래픽 통계정보를 수집하기 위해서 그외의 프로토콜 정보도 포트번호가 0번인 특별 플로우에 저장된다. ICMP 패킷의 경우 포트번호는 ICMP 타입과 코드로 대체된다. 한에이전트가 여러 개의 회선을 측정할 수 있지만 먼저 에이전트 플랫폼의 성능 및 수집 회선의 트래픽 양을 함께 고려해야 된다. 먼저 분석 서버는 플로우와 패킷 레코드를 출발지 및 목적지 포트번호를 바탕으로 4가지 유형중 하나로 분류해야한다. 그 다음 패킷 컨텐츠와 기타 메핑정보를 이용하여 구체적인 응용으로 인식이 이루어진다. 마지막으로 chain based rule matching 알고리즘에 의해서 응용의 서브 그룹으로 세분화 되어진다. 서버는 여러 에이전트로부터 전송된 레코드를 통합해서 종합적으로 처리할 수 있다. 인터넷의 비대칭적인 라우팅의 특성상, Type-DP 플로우는 유발플로우가 발견된 회선이 아닌 타 회선 혹은 회선들에서 발생할 수 가 있다. 마찬가지로 type-RR 플로우도 원래 플로우인 type-PI 플로우가 존재하는 회선이 아닌 곳에서 나타날 수 있다. 이러한 가능성 때문에 서버는 복수개의 에이전트의 레코드를 관리를 해야만 한다. 인식된 결과는 데이터베이스에 저장되며 GUI를 통해서 쉽게 접근 및 표현이 가능하다. 한편 ARCL 파일은 에이전트와 서버가 응용인식을 위해 필요한 위에서 언급된 모든 정보들을 담고 있다.

에이전트는 실시간으로 운용될 수 있도록 설계가 되어 있으며 이를 지원할 수 있는 성능의 기준이 수집 카드의 실시간 처리 능력에 의존적이다. 앞에서 언급되었듯이 과금분야로 본 시스템의 응용분야를 국한할 경우 서버는 실시간이 아닌 batch 방식으로 운용되어도 무리가 없으나 실시간으로 처리도 가능하다. 그러나 OC 48 혹은 OC 192와 같은 초고속 회선의 경우에는 에이전트 및 서버에 많은 부하가 걸릴 수 있기 때문에 보다 정교한 운용이 필요하며 이 부분에 대한 연구는 향후 과제로 처리할 예정이다.

에이전트

<그림 3>은 에이전트의 내부 구조 및 처리과정을 상세히 묘사하고 있다. 에이전트는 패킷 캡쳐 (Packet Capturer (PC))와 패킷 프로세서 (Packet Processor (PP)) 두 모듈로 구성된다. 이 두 모듈이 함께 하나의 논리적인 회선을 담당하며, 한 에이전트 시스템에서 다수개의 PC, PP 쌍의 인스턴트를 동작시킬수 있다. PC의 주요역할은 원시 IP 패킷을 커널메모리에서 사용자 메모리로 가능한 손실이 없이 복사하는 것이다. PP로 전달된 패킷은 먼저 조각 처리 루틴에서 필요시 조각을 모아 원래의 패킷으로 복원을한다. 그다음 플로우 조립을 시작하는데 이때 플로우 캐쉬 루틴이 개입하며 플로우 첫번째 패킷이 도착하면 새로운 플로우 캐쉬 엔트리를 생성하고 동일 플로우에 속하는 패킷들이 도착하면 계속 이 엔트리에 업데이트 한다. 그리고 플로우 종료 루틴이 모든 기준 시간마다 실행되며 active, inactive, 혹은 TCP FIN timeout에 의해서 종료된다. 패킷별 처리 루틴의 마지막 과정으로 payload 검색을 거치게 된다. 이 과정에서는 type-PI 및 type-DP 유형의 분류를 위한 응용식별자 검색이 이루어진다. 서버로 전송되는

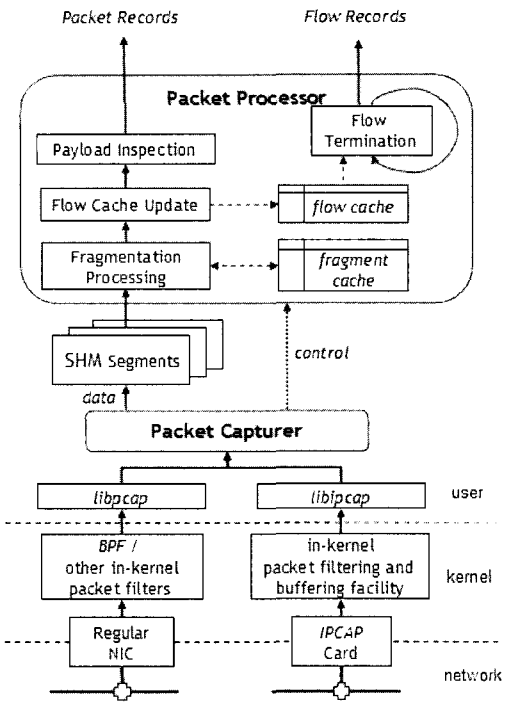


그림 3. 에이전트의 내부 구조 및 처리 절차도
Fig. 3. Agent Architecture.

레코드의 양을 줄이기 위해 식별자가 포함된 payload만 포함하게 된다. 손실없는 실시간 수집을 위해서 다양한 속도를 지원하는 캡쳐 카드(DS 3, Fast- Ethernet, OC 3/12 POS/ATM, GigaEthernet)인 IPCAP 카드를 직접 설계하고 제작하였으나 본고에서는 제한된 공간으로 상세내용은 생략하기로 한다. 그러나 에이전트가 BPF와 libpcap[10]상에서 동작할 수 있도록 설계되었기 때문에 이 기능을 지원하는 UNIX 운용체제를 탑재한 PC 혹은 워크스테이션이면 실행되는데 문제가 없다. 주로 OC 12 이하의 저속 회선에서는 이렇게 PCAP만 사용해도 큰 패킷 손실없이 측정 및 분석이 가능하다.

분석서버

<그림 4>는 분석서버의 내부 구조 및 분석 절차를 보여준다. 응용 인식 및 분석은 크게 전처리 및 후처리 과정으로 나뉘어 수행된다. 다수개의 회선으로부터 레코드파일이 수집되었을 경우 각파일별로 전처리과정이 호출되지만 후처리과정은 레코드 파일 전송 주기인 서버 동작 주기별로 전처리과정이 끝난 모든 레코드 파일들을 동시에 분석한다. Type-FP 혹은 type-PI 플로우는 전 플로우의 정보가 필요없기 때문에 첫번째 플로우 분류과정에서 식별이 된다. 이러한 플로우는 즉시 port_rep_name으로 메핑이 되며 이 정보는 영구적으로 자체 응용테이블에 갱신된다. 이러한 플로우에 속한 패킷들도 ARCL 파일에 정의된 메칭 규칙을 기준으로 해당 decision_group으로 분류가 이루어진다. 한편, 전처리 과정 중 type-DP 및 type-RR 플로우를 유발하는 플로우를 찾을수 있으며 후처리시 활용하기 위하여 DP ref_table 및 RR_ref_table에 각각 참조정보를 저장해 둔다. 참조정보 테이블은 모든 회선에 공통적으로 적용된다. 한편 후보 type-RR 플로우는 100%는 아니지만 어느정도의 확실성을 가지고 인식이 가능하기 때문에 RR_candidate 테이블에 정보를 저장해 둔다. 그리고 나머지 모든 플로우는 DP_candidate 파일에 저장해 둔다. 이 중 일부는 인식이 되지 않는 미 식별 플로우가 될수도 있다. 한 플로우가 장시간 존재하는 플로우의 첫번째 서브플로우인 경우 그 플로우의 분류결과는 연속플로우 테이블에 저장되는데 이후에 연속되는 플로우들이 동일한 과정을 반복하지 않게 하기 위함이다.

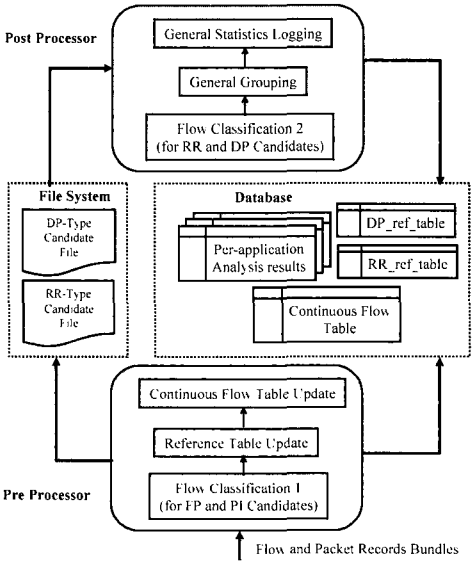


그림 4. 분석서버의 내부구조 및 처리 절차도
Fig. 4. Analysis Server Architecture.

주어진 처리 주기내에 모든 레코드 파일 분석을 위한 전처리 과정이 종료되었을 경우 후처리 과정을 거치게 된다. 후처리과정의 첫번째 임무는 RR_candidate 및 DP_candidate 파일에 저장된 플로우들을 분류하는 것이다. RR_candidate 파일 중 한 플로우가 RR_ref_table의 참조정보와 일치되었을 경우 그 플로우는 type-RR 플

로우로 확정되며 따라서 port_rep_name으로 메핑된다. 비슷하게 DP_candidate 파일에 있는 플로우들도 처리된다. 확실히 분류가 완료된 플로우 정보들은 해당 응용테이블에 영구히 갱신되며 관련 부속 패킷들도 확실히 분류가 된다. 이러한 절차를 거쳐 플로우 및 패킷 식별 과정이 끝난다. 나머지 부속 과정들은 국가와 AS 기반 그룹핑 혹은 회선 혹은 회선 세트기반 그룹핑과 같은 다양한 통계정보를 분석하기 위함이다.

플로우를 port_rep_name으로 메핑하는 것은 두가지 과정으로 구성된 인식 도출 알고리즘에 의해서 이루어진다. 첫번째 과정은 한 플로우를 한 혹은 두개의 포트로 메핑하는 것이며 두번째는 포트를 port_rep_name으로 메핑하는 것이다. <그림 5>는 두 과정상 주요 아이디어를 부연하는 플로우차트이다. 첫번째 해결과정의 주 아이디어는 잘알려진 포트번호는 등록포트번호에 비해서 영향력면에서 더 높은 우선권을 가진다는 것이다. 이 가정은 직관적인데 왜냐하면 한 플로우의 포트번호가 잘 알려진 포트 번호 범위에 속하고 다른 플로우가 등록포트번호 범위에 속할 경우 전자가 유효 서비스 포트가 되고 후자는 epheneral 포트가 된다. 그리고 두번째 해결과정의 주요 포인트는 인식유형간에 우선순위가 type-PI, type-FP, 그리고 type-RR 순으로 낮아진다는 것이다. Type-DP 플로우는 타 유형과 우선순위의 영향을 받지 않는다.

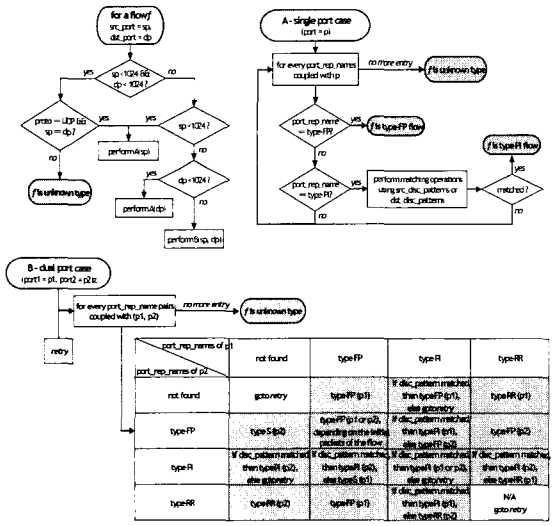


그림 5. 포트 및 유형 식별 및 분류를 위한 플로우 차트
Fig. 5. Flow chart to recognize port and application type.

V. 시스템 구현 및 성능

본고에서 제안된 아이디어와 구조는 Wise*TrafView라는 시스템으로 구현되었다. 이 시스템은 ETRInet, 포항공대 등에 설치되어 시험 운용중에 있다.

ETRInet에서는 두 ISP와 인터넷으로 연결되는 DS3 회선에 설치되어 있다. 이 사이트에는 2500여명의 사용자가 있으며 수천대의 컴퓨터와 네트워크 장비들이 사용되고 있다. Incoming 및 Outgoing 회선으로부터 수집된 패킷들이 에이전트로 전달된다. 분석결과 평균 대역폭, 패킷전송량, 플로우수가 46.52 Mbps이며, 5.325 Kpps, 174.1 fps 최대값들은 148.23 Mbps, 18.242 Kpps, 1.359 Kfps이다. 두 논리 회선(incoming & outgoing)을 측정하기위해서 에이전트 시스템에서는 두 인스턴스의 PC 및 PP 쌍을 실행시키고 있다. 에이전트 및 서버 플랫폼은 2개의 Zeon CPU, 2GB 메인메모리, 66MHz 64 bit PCI 버스의 사양을 가진 PC-서버에서 실행되고 있다.

PC와 PP에서는 패킷 손실이 없이 실시간으로 처리가 되고 있다. 즉 PC 및 PP에서 처리하는 속도가 패킷을 수집하는 속도를 능가하고 있다. ETRInet의 경우 PP가 하나의 패킷 레코드 파일 (이하, 패킷번들)을 처리하는데 걸리는 평균시간은 1.782초이다. 그리고 PC가 패킷을 수집하고 패킷번들로 만드는데 걸리는 시간은 31.006 초이다. 단순한 계산만으로도 현재 운용중인 에이전트 시스템의 경우 대략 92.65 (5.325 * 31.006/1.782) Kpps 및 809.43(46.52 * 31.006/1.782) Mbps의 성능을 지원할 수 있는 것을 확인 가능하다. 이러한 추정치와는 상관없이 본 시스템은 다양한 방법으로 과부하 상황을 피할 수 있다. 예를들면, type-PI 혹은 type-DP 플로우의 비율을 조정함으로써 에이전트 및 서버의 부하를 낮출 수있다. 따라서 초고속 회선의 경우 모든 플로우를 type-FP로만 설정할 경우 Netlfow와 같이 일반적인 플로우기반의 측정시스템과 동일한 성능을 가질수 있다. 다만 이 경우 응용인식율이 낮아지는 것은 감수를 해야된다.

Postech은 인터넷 연결을 위해서 100Mbps 메트로 이더넷을 통해서 ISP와 연결이 되어있다. 이 회선에 지난 5월 중 2주간의 측정 및 분석을 시도하였다. <그림 6>은 일반적인 응용 인식 방식과 본고에서 제안한 방식의 차이점을 비교 설명하고 있다. 주목할 점은 본제안 시스

템의 분석데이터를 보면 포트80번을 사용하는 HTTP 트래픽양이 11.8% 줄어든것을 볼 수있는데 이유는 타 응용이 파이어월을 피하기 위해 이 포트를 사용하기 때문인 것으로 추정된다. Passive FTP 트래픽이 6GB나 차지하고 있으나 본 시스템이 아니면 측정이 되지 않고 미확인 응용으로 인식이 되었을 것이다. 마지막으로 포트 5003을 사용하는 응용도 미확인 혹은 특정 설정응용(만약 이 포트가 특정응용으로 설정되어있었다면)으로만 인식이 되었을것이지만 본제안 시스템에서는 이 포트가 Peer to Peer 응용 및 HTTP로 다양하게 사용되고 있음을 확인할 수 있다.

PosTech Traffic Breakdown

Port/Application	Port-based Accounting	Contents-based Accounting
80/HTTP	67 GB	59.1 GB (11.8% reduced)
21/FTP_CTRL	0.29 GB	0.28 GB
20/FTP_DATA	43 GB	42 GB
?/FTP_DATA_PASSIVE	n/a	6 GB (14.3% of FTP_DATA, 2% of the total volume)
5003/?	692 MB	HTTP: 13.2 MB
		BUGS_MUSIC: 420.8 MB
		EDONKEY: 172.3 MB
		etc.: 85.7 MB

- PosTech Campus Network
(24h sum in May, 304GB total volume)

그림 6. 포항공대 응용트래픽 분류
Fig. 6. Postech Application Traffic Classification.

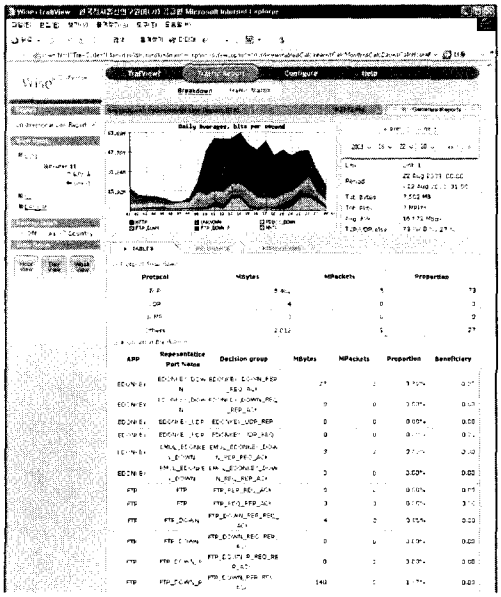


그림 7. Wise*TrafView 시스템 사용자 인터페이스
Fig. 7. Wise*TrafView GUI.

<그림 7>은 본 제안 시스템의 GUI의 한 스냅샷을 보여준다. 화면의 왼쪽 프레임에는 수집 및 분석하고자 하는 시스템 및 회선에 대한 정보가 있고 오른쪽 프레임에는 분석된 응용중 트래픽양이 가장많은 대표 6개의 응용에 대한 요약 정보 및 수집기간, 총수집량, 수집 패킷수, 평균 대역폭 및 프로토콜별 트래픽양등을 나타내는 요약표, 마지막으로 각 응용별 및 서브그룹별 상세 트래픽 통계정보를 나타내는 표로 구성된다.

VI. 결론 및 향후 연구

현재 인터넷 응용들의 동적인 특성으로 인해, 인터넷에서 응용 트래픽의 정확한 사용량 산정은 패킷 컨텐츠 검색, 플로우기반 분석, 서브 트랜잭션 플로우간의 상관관계 분석, 및 실시간 패킷 수집 성능 등이 적절하게 조화가 되어야 가능한 것이다. 본고에서는 이러한 문제의 해결방안으로 여러가지 아이디어를 바탕으로 독창적인 제안을 하였다.

또한 이러한 기술적인 아이디어 및 구조설계를 바탕으로 Wise*TrafView 시스템을 구현하였으며 웹기반의 사용자 인터페이스를 통해 쉽고 간결하게 분석된 트래픽정보를 표현할 수 있게 하였으며, 몇 군데 주요 현장에 설치하여 시험 운영중에 있다. 현재까지의 시험결과에 대체로 만족하고 있으며 보다 상세한 분석 결과를 취합하고 있는 과정에 있다. 또한 본 시스템을 국내뿐만 아니라 국제적으로 적용하기위한 노력으로 APEC 산하 프로젝트[11]를 2003년 1월경에 수주하였으며 그 액션항목 중 하나로 9월경 중국에 1차 설치 및 시험운용이 계획되어있다. 이외에 순차적으로 호주, 싱가폴 등 APEC 국가들과의 긴밀한 협조가 계속이루어질 전망이다.

현재까지 본시스템을 OC 3 속도까지 시험을 하였으며 속도를 OC 12 및 Gbps로 높이는 연구를 진행중에 있다. 2003년 12월까지는 기본적인 프로토타입은 개발될 것으로 예상된다. 그리고 본시스템의 응용분야를 주로 과금에 맞추었지만 트래픽 프로파일링, 트래픽 엔지니어링, 사이버 공격 탐지 등과 같은 타분야로의 활용이 가능한 만큼 부가기능 연구 및 개발에 적극적인 대처를 할 계획이다.

참 고 문 헌

[1] Colleen Shannon, David Moore, and k claffy. "Characteristics of Fragmented IP Traffic on Internet Links", Proc. of ACM SIGCOMM Internet Measurement Workshop, San Francisco, USA, Nov. 2001.

[2] CIADA's OCxMon & NetTraMet. http://www.caida.org/tools/.

[3] TCPDUMP. http://sourceforge.net/projects/tcpdump/

[4] Ethereal. http://www.ethereal.com/.

[5] Sprint ATL, "IP Monitoring Project," http://www.sprintlabs.com/Department/IP-Interworking/Monitor/.

[6] http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm.

[7] K. Keys, D. Moore, Y. Koga, E. Lagache, M. Tesch, and K. Claffy, "The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite," Proc. of Passive and Active Measurement Workshop 2001, Amsterdam, Netherlands, April 2001.

[8] D. Plonka, Flowscan: A network traffic flow reporting and visualization tool. In Proceedings of USENIX LISA, 2000.

[9] Cisco NBAR. http://www.cisco.com/warp/public/732/ Tech/qos/nbar/

[10] Steven McCanne, Van Jacobson. The BSD Packet Filter : A New Architecture for User level Packet Capture, In Proc. of the Winter 1993 USENIX Conference, 259 269. USENIX Association, Jan 1993.

[11] ETRI. APEC TEL 01 2003, Flow based Internet Traffic Measurement and Analysis, APEC TEL, December 2002.

저 자 소 개

崔 台 相(正會員)

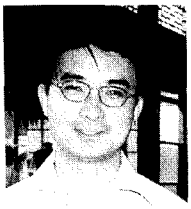
1995년 12월 : 미주리-캔사스 주립대학 컴퓨터통신학과 공학박사. 1996년 4월~1998년 12월 : ETRI 멀티미디어통신팀에서 "고품질 VOD 시스템" 개발. 1999년 1월~2001년 12월 : ETRI 인터넷구조팀에서 "MPLS 기반 트래픽엔지니어링 서버" 개발. 2002년 1월~현재 : ETRI 인터넷트래픽관리팀에서 "컨텐츠 인식기반 인터넷 응용 트래픽 수집 및 분석 시스템" 개발. <주관심분야 : 인터넷 트래픽 엔지니어링, QoS 관리, 트래픽 분석>

朴 貞 淑(正會員)

2001년 2월 : 대구가톨릭대학교 전산통계학과 이학박사. 2001년 2월~2001년 12월 : ETRI ETRI 인터넷구조팀에서 "MPLS 기반 트래픽엔지니어링 서버" 개발. 2002년 1월~현재 : ETRI 인터넷트래픽관리팀에서 "컨텐츠 인식기반 인터넷 응용 트래픽 수집 및 분석 시스템" 개발. <주관심분야 : 인터넷 트래픽 엔지니어링, 트래픽 측정, 성능 분석, 트래픽 모델링.>

尹 承 鉉(正會員)

1997년 2월 : 성균관대학교 산업공학과 공학박사. 1997년11월~1998년 12월 : ETRI NTB 팀에서 네트워크 시뮬레이션 연구. 1999년 1월~2001년 12월 : ETRI 인터넷구조팀에서 "MPLS 기반 트래픽엔지니어링 서버" 개발. 2000년 6월~2001년 5월 : "국가 인터넷 통계 수집 시스템을 위한 인터넷 트래픽 측정시스템" 개발. 2002년 1월~현재 : ETRI 인터넷트래픽관리팀에서 "국제회선 인터넷 요금 분담을 위한 인터넷 응용 트래픽 수집 및 분석 시스템" 개발. <주관심분야 : 인터넷 트래픽 엔지니어링, 트래픽 측정 및 분석, 망 설계, 시스템 성능분석>

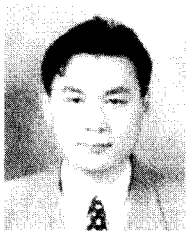


金 亨 煥(正會員)

2000년 2월 : 충남대 컴퓨터과학과 석사. 1991년 2월 : 한양대 전자공학과 학사. 1991년 1월~2002년 1월 : ETRI 실시간OS팀에서 TDX-10, ACE-256/2000 교환기용 "실시간 운영체제" 개발. 2001년 3월~2002년 2월 : ETRI 실시간OS팀에서"테라라우터 제어계" 개발. 2002년 1월~현재 : ETRI 인터넷트래픽관리팀에서 "인터넷 회선요금 공정분담을 위한 트래픽 측정기술" 개발. <주관심분야 : 인터넷 트래픽 엔지니어링, QoS관리, 내장형 실시간시스템 소프트웨어기술>

金 昶 勳(正會員)

1999년 2월 : 서울대학교 컴퓨터공학과 석사. 1999년 3월~2001년 12월 : ETRI 인터넷구조팀에서 "MPLS 기반 트래픽엔지니어링 서버" 개발. 2002년 1월~현재 : ETRI 인터넷트래픽관리팀에서 "응용 인식가능한 설정형 인터넷 응용 트래픽 수집 및 분석 시스템" 개발. <주관심분야 : 인터넷 운용, 인터넷 트래픽 측정 및 관리, 차세대 인터넷 응용 등.>



鄭 炯 錫(正會員)

2000년 2월 : 광운대학교 전자통신공학과 공학박사. 1999년 11월~2001년 12월 : ETRI 인터넷구조팀. "DiffServ기반 QoS 설정 서버" 개발, "MPLS 트래픽엔지니어링 서버" 개발. 2002년 1월~현재 : ETRI 인터넷트래픽관리팀에서 "컨텐츠 기반 인터넷 응용 트래픽 수집 및 분석 시스템" 개발. <주관심분야 : 차세대 망관리 기술, 인터넷 트래픽 분석>

李 秉 俊(正會員)

1998년 : 서울대학교 컴퓨터공학과 공학석사. 2001년 12월~현재 : ETRI 인터넷트래픽관리팀에서 "인터넷망 관리를 위한 Wise<NT> 시스템" 및 "컨텐츠 인식기반 인터넷 응용트래픽 수집 및 분석 시스템 개발". <주관심분야 : 인터넷 트래핑 엔지니어링, 트래픽 측정 및 분석>

鄭 泰 洙(正會員)

1983년 2월 : 경북대학 전자공학과 공학석사. 1983년 3월 : ETRI 데이터통신연구실, 통신망구조연구실 연구원, 광대역통신방식연구실, 광대역프로토콜연구실 실장, 체계종합팀, 인터넷구조팀 팀장. 현재 : 인터넷트래픽관리팀 팀장, 관심분야는 통신망기술, 인터넷 트래픽 엔지니어링, QoS 관리