논문 14-39B-04-01
The Journal of Korea Information and Communications Society '14-04 Vol.39B No.04
http://dx.doi.org/10.7840/kics.2014.39B.4.191

시그니쳐 계층 구조에 기반한 HTTP 트래픽 분석 시스템의 처리 속도 향상

최 지 혁◆, 박 준 상*, 김 명 섭°

Processing Speed Improvement of HTTP Traffic Classification Based on Hierarchical Structure of Signature

Ji-Hyeok Choi◆, Jun-Sang Park*, Myung-Sup Kim°

요 약

최근 웹 기반의 다양한 응용과 서비스의 제공으로 인해 HTTP 트래픽의 양이 급격하게 증가하고 있다. 따라서 안정적인 네트워크 관리를 위해서 HTTP 트래픽에 대한 분석이 필수적으로 요구된다. HTTP 트래픽을 다양한 관점에서 분석하기 위해서는 다양한 시그니쳐 기반 분석 방법 중에 페이로드 시그니쳐 기반 분석 방법이 효과적이다. 하지만 트래픽 분류 있어서 페이로드 시그니쳐 기반 방법은 고속 링크의 대용량 트래픽을 실시간으로 처리하는 과정에서 헤더 정보 및 통계 정보 이용 방법론에 비해 상대적으로 높은 부하를 발생시키며 처리 속도가 느린 단점을 갖는다. 따라서 본 논문에서는 HTTP 시그니쳐의 계층 구조에 기반하여 HTTP 트래픽을 다양하게 분류할 수 있는 방법론을 제시한다. 또한 계층 구조의 특징을 반영하여 패턴 매칭의 처리 속도 향상을 위한 방법을 제안한다. 제안하는 방법을 학내망의 실제 트래픽에 적용하여 평가한 결과, Aho-Corasick 알고리즘 보다 더 빠른 처리 속도를 보일 수 있었다.

Key Words : HTTP traffic classification, processing speed, hash, signature hierarchy

ABSTRACT

Currently, HTTP traffic has been developed rapidly due to appearance of various applications and services based web. Accordingly, HTTP Traffic classification is necessary to effective network management. Among the various signature-based method, Payload signature-based classification method is effective to analyze various aspects of HTTP traffic. However, the payload signature-based method has a significant drawback in high-speed network environment due to the slow processing speed than other classification methods such as header, statistic signature-based. Therefore, we proposed various classification method of HTTP Traffic based HTTP signatures of hierarchical structure and to improve pattern matching speed reflect the hierarchical structure features. The proposed method achieved more performance than aho-corasick to applying real campus network traffic.

※ 본 연구는 BK21플러스 사업의 지원을 받아 수행된 결과임.
◆ First Author : ETRI, jihyeok_choi@etri.re.kr, 정회원
° Corresponding Author : Korea University Network Management Lab. Dept. of Computer and Information Science, tmskim@korea.ac.kr, 정회원
* Korea University Network Management Lab. Dept. of Computer and Information Science, junsang_park@korea.ac.kr, 학생회원
논문번호 : KICS2013-12-534, Received December 12, 2013; Revised January 3, 2014; Accepted April 21, 2014

Ⅰ. 서 론

HTTP 프로토콜을 사용하는 웹 기반의 서비스와 응용 프로그램이 급격히 증가하고 있는 추세이다. 전통적인 웹 기반 서비스뿐만 아니라, 특정 P2P 서비스는 네트워크 방화벽에 의한 차단을 피하기 위하여 HTTP를 사용하고 있다. 이와 같은 현실 속에서 네트워크의 원활한 서비스를 제공하기 위해서는 응용 계층 프로토콜 단위가 아닌 응용 서비스 단위 분석이 중요하게 되었다. 트래픽 엔지니어링, QoS 정책 수립, SLA(Service Level Agreement) 등의 네트워크 관리 정책의 신뢰성을 보장하기 위해서는 다양한 종류의 HTTP 기반 트래픽을 응용 서비스 단위로 정확하게 분류할 수 있는 방법이 요구된다[1,2].

HTTP 트래픽을 응용 서비스 별로 정확하게 분석하고, 동시에 높은 분석률을 보장하기 위해서는 페이로드 시그니쳐 기반 분석 방법이 적합하다[3-6]. 하지만 분류 시스템의 처리 속도에 있어 현재의 고속 네트워크 상에서 발생하는 대용량 트래픽을 실시간으로 처리하기에 부적합한 방법이다. HTTP 프로토콜에 기반한 응용 서비스의 종류가 다양해지고, 대용량의 트래픽을 발생시키는 서비스의 사용이 증가하고 있는 추세를 고려했을 때 페이로드 기반 분석 방법의 처리 속도 문제는 반드시 해결되어야 하는 과제이다.

본 논문에서는 HTTP트래픽 분석 결과의 다양한 활용 목적을 고려하여 다각적인 트래픽 분류 기준을 제시한다[7,8]. 제안한 분류 기준에 부합하는 분류 결과를 도출하기 위해서 HTTP프로토콜의 개별 필드를 계층화하여 시그니쳐 모델을 정의한다. 계층적 구조의 시그니쳐 모델은 다각적 분석이 가능하며, 시그니쳐의 탐색 공간을 최소화하여 분류 시스템의 처리 속도 향상 시킬 수 있다. 또한 시그니쳐의 각 계층을 해시 키 기반으로 매칭하여 분류 시스템의 처리 속도를 향상시킬 수 있는 방법론을 제안한다. 제안하는 방법론을 학내망의 실제 트래픽에 적용하여 처리 속도를 분석한 결과, 문자열 매칭에 많이 사용되고 있는 Aho-Corasick 알고리즘보다 제안하는 방법이 더 빠른 처리 속도를 보이는 것을 확인하였다.

본 논문의 구성은 다음과 같다. 서론에 이어, 2장에서는 관련연구에 대해 기술하고, 3장에서는 제안하는 방법론을 기술하고, 4장에서는 제안하는 방법을 학내 망에 적용하여 그 타당성을 증명한다. 마지막으로 5장에서는 결론 및 향후 연구에 대해 기술한다.

Ⅱ. 관련 연구

웹 기반의 다양한 응용과 서비스의 제공으로 인해 HTTP 트래픽의 양이 증가하고 있는 추세이지만 기존의 응용 레벨 트래픽 분석 연구는 응용 계층 프로토콜 기준의 분석이 주를 이루고 있다. Garcia-Dorado *et. al*은 ISP와 캠퍼스 네트워크에서 발생하는 트래픽을 대상을 트래픽 트렌드을 분석하였다[9]. ISP와 캠퍼스 네트워크 모두 30% 이상의 HTTP 트래픽을 사용하고 하고 있을 것을 알 수 있다. 따라서 HTTP 트래픽을 세분화된 서비스 단위로 분석하여 네트워크 관리를 위한 활용도 높은 결과를 제시하여야 한다.

HTTP 트래픽이 점유하는 비율이 높아짐에 따라서 최근에는 HTTP 트래픽을 세분화하여 분류하는 연구가 진행되고 있으나 비디오, 오디오와 같은 단순한 트래픽 유형 별로 분류하는 방법론이 제안되었다. Samruay K. *et. al*은 HTTP 트래픽을 HTTP Content-type필드의 시그니쳐와 패킷의 통계 정보를 이용하여 분석하는 방법을 제안하였다. 또한 Tomasz B. *et. al*은 머신 러닝 기법을 이용하여 HTTP 트래픽을 분류하는 기법을 제안하였다[10]. 하지만 제안하는 방법은 HTTP를 통해 전송되는 비디오와 오디오 등과 같은 트래픽의 유형 별로 분석하여 결과의 활용도가 부족하다. 현재의 HTTP 기반 트래픽은 HTML 문서의 전송뿐만 아니라 파일 전송, 스트리밍 등의 다양한 목적으로 활용되고 있기 때문에 세부적인 서비스 별 분류 결과를 제공해야 한다.

다양한 분석 결과를 얻기 위해서는 그에 맞는 시그니쳐 모델이 정의 되야 한다. 하지만 기존의 시그니쳐 모델들은 분석하고자 하는 트래픽의 특성에 맞게 기술되어 있지 않고, 분석 알고리즘에 맞게 정의 된 경우가 많았다[11]. 따라서 동일한 시그니쳐를 다른 분석 장비에 적용하면 동일한 성능을 보장할 수 없는 문제가 발생한다. 따라서 페이로드 시그니쳐 모델은 분석 대상이 되는 트래픽의 특징을 반영하여 기술 되야 하고, 매칭 알고리즘 또한 트래픽 특징에 맞게 수정되어야 한다.

HTTP 트래픽을 빠르고 정확하게 분류하기 위하여 여러 가지의 방법이 존재하는데 패킷의 페이로드까지 정밀하게 검사해야 높은 정확도의 분류가 가능하다. 페이로드 시그니쳐 기반 분석에 사용되는 도구로 오픈 소스 기반 시스템인 Snort가 널리 사용되고 있다[12]. Snort의 모듈 단위 수행 시간을 비교한 결과, 70%의 시간이 패턴 매칭 과정에서 소모되었다. Snort 에서는 패턴 매칭 알고리즘으로 Aho-Corasick을 사용한

다[13,14]. Aho-Corasick은 멀티플 패턴 매칭에 기반한 알고리즘으로 복수 개의 시그니쳐를 하나의 트리로 구성하여 한번의 트리 탐색으로 분류가 가능하기 때문에 시그니쳐에 대한 탐색 공간을 줄일 수 있다. 하지만 이러한 방법은 HTTP 트래픽을 응용 서비스로 분류하기 위한 트래픽의 특성을 반영하지 않기 때문에 제한적인 성능 향상을 나타낸다.

HTTP 트래픽의 다양성을 고려하여 세부적인 서비스 별로 정확하게 분류할 수 있는 방법론과 많은 양의 시그니쳐를 빠르게 분석할 수 있는 기법에 대한 연구가 요구된다. HTTP 트래픽 분석 시스템의 처리 속도를 향상 시키기 위해서는 매칭 알고리즘의 성능 개선이 필요하다. 따라서 본 논문에서는 HTTP 트래픽을 서비스 별로 분석할 수 있는 페이로드 시그니쳐를 정의하고, 시그니쳐를 계층화하여 시그니쳐 간의 중복성을 제거하고, 탐색 공간을 줄여 줌으로써 처리 속도를 향상 시킨다. 또한 트래픽과 매칭시 해시 구조를 사용함으로써, 하나의 트래픽을 여러 개의 시그니쳐와 매칭하지 않고, 하나의 시그니쳐와 매칭할 수 있게 하였다.

Ⅲ. 제안하는 방법

본 장에서는 HTTP 트래픽을 응용 서비스 별로 분석할 수 있는 방법과 처리 속도를 극대화할 수 있는 구현 방법에 대해 기술한다.

3.1 HTTP 트래픽 분류 방법

본 절에서는 HTTP 트래픽을 세분화하여 분석하기 위한 분류 기준, 시그니쳐, 분류 방법을 기술한다.

HTTP 트래픽을 응용 서비스 별로 다양한 관점으로 분석하기 위해서는 트래픽 분류 기준이 필요하다. 본 논문에서 제안하는 분류 기준은 서비스, 응용, 기능 측면의 다각적 분석 결과를 제공함으로써 트래픽의 이해도와 활용도를 높일 수 있다. 표 1은 본 논문에서 제안하는 트래픽 분류 기준과 각 기준에 대한 정의를 보여 준다.

표 1. 분류 기준
Table 1. Classification criteria

Criterion	Definition
Service	사용자에게 content를 제공되기 위해 개발된 모든 형태의 IT서비스
Application	서비스를 이용하기 위해 Client에서 사용된 응용 프로그램
Function	사용자 행동 기준에 따른 트래픽의 발생 목적

본 논문에서는 플로우 기반 트래픽 분석을 수행하며, 1개의 플로우는 서비스, 응용, 기능 측면으로 분류할 수 있다. 각 분류 기준의 필요성은 기존의 연구를 통해 그 타당성을 확인하였다[7, 8]. 이러한 분석 결과는 더욱 정확하고 세밀한 분석 결과를 제공할 수 있다. 예를 들어, 사용자가 Chrome 브라우저를 사용하여 youtube 멀티미디어 사이트에서 뮤직 비디오를 감상 하는 행위을 했다고 가정한다. 1개의 플로우는 youtube 서비스, chrome 응용, video_streaming 기능으로 분석될 수 있다. 이처럼 다차원적 분석은 1개의 플로우를 3가지 관점으로 분석함으로써 트래픽에 대한 사용자의 이해를 넓힐 수 있고, 다양한 목적의 트래픽 분석에 폭 넓게 사용될 수 있다. 예를 들어, 특정 브라우저에서 악성 코드가 유포되고 있는 상황이라면 해당 브라우져에서 발생하는 트래픽을 차단할 수 있고, youtube 서비스의 video_streaming 기능에 의해서 트래픽이 과다하게 발생하여 업무용 트래픽에 영향을 미치는 경우에는 해당 서비스의 기능 단위로 트래픽을 제어할 수 있다.

1개의 HTTP 플로우를 3가지 분류 기준으로 분석하기 위해서는 각 분류 기준을 구별할 수 있는 정보를 트래픽으로부터 추출할 수 있어야 한다. HTTP 프로토콜에서 정의하고 있는 다양한 필드 중 User-Agent, Host, Uri 필드는 3가지 분류 기준을 식별할 수 있는 정보를 포함하고 있다. 그림 1는 토렌트를 사용할 때 발생하는 HTTP 패킷의 첫 요청 패킷과 응답 패킷의 내용을 보여준다.

Uri는 요청하는 자료의 경로와 요청에 필요한 속성

Request Packet
GET /checkupdate.php?cl=&v=103246420 HTTP/1.1 Host : update.utorrent.com:7070 User-Agent : BTWebClient/3130(27220) AcceptEncoding : gzip Connection : Close
Response Packet
HTTP/1.1 200 OK Server : nginx/1.4.1 Date : Fri, 09 Aug 2013 01:54:16 GMT Content-Type : text/html Content-Length : 1459 Connection : close X-Powered-By : PHP/5.4.16 Expires : Thu, 21 Jul 1980 00:00:00 GMT Cache-Control : private Last-Modified : Fri, 09 Aug 2013 01:54:16 GMT

그림 1. HTTP 요청/응답 패킷 페이로드
Fig. 1. HTTP request/reply packet payload

과 속성 값을 기술하는 필드이다. 따라서 HTTP 트래픽의 기능을 식별할 수 있는 정보를 포함하고 있다. 그림 1의 Uri 필드에 checkupdate라는 키워드를 통해 업데이트 기능임을 알 수 있다. 그림 1과 같이 HTTP의 호스트 필드는 도메인명(utorrent.com)과 호스트명(update)으로 구분할 수 있다. 도메인은 분류 기준 중에 서비스의 이름을 식별하기 위한 시그니쳐 사용되며, 호스트 필드는 해당 도메인에서 제공하는 기능을 나타낸다. User-Agent 필드는 3 가지 분류 기준 중에 클라이언트에서 사용하는 응용 프로그램의 정보를 명시해 주는 필드이다. 그림 1에서 User-Agent는 BTWebClient로 utorrent 서비스를 사용할 때 나타나는 클라이언트 프로그램이다. HTTP의 Uri, 호스트, User-Agent 필드는 Uri, 호스트, 도메인, User-Agent로 구분되어 서비스, 응용, 기능의 분류 기준에 적합한 시그니쳐 형태로 기술 된다. 네 가지 필드로 이루어진 시그니쳐 포맷은 다음과 같다.

<signature code="1">
<msg="utorrent - BTWebClient - update">
<payload = User-Agent : BTWebClient, Domain : utorrent, Host : update, Uri: checkupdate>

시그니쳐 코드는 시그니쳐에 대한 식별자이며, msg는 3 가지 분류 기준에 해당하는 서비스, 응용, 기능을 명시한다. 페이로드는 User-Agent, 도메인, 호스트, Uri에 해당하는 시그니쳐를 명시한다. 3가지 분류 기준에 모두 해당하는 경우에 시그니쳐로 등록 된다. 예외적으로 시그니쳐를 추출하는 정답지 트래픽에서 4가지 필드를 모두 추출할 수 없는 경우에는 User-Agent와 도메인 중 1개만이도 추출 가능하다면 시그니쳐 인정한다. User-Agent나 도메인이 누락되는 트래픽의 양은 미미하지만 이러한 트래픽 또한 분석 대상으로 포함하여 서비스 또는 응용으로 식별하는 것을 목표로 한다.

필드 단위로 시그니쳐를 추출하고 분석하기 때문에 정확한 분류 결과를 도출하기 위해서는 각 필드의 시그니쳐를 추출하는 시점과 분류 시점에 동일한 필드 추출 방법이 요구된다. 그림 2는 그림 1의 요청 패킷에서 4 개 필드를 추출하는 방법을 보여준다. 시그니쳐를 추출하는 대상 패킷은 플로우의 첫 번째 요청 패킷이다. 이로 인해 플로우의 통계 정보(e.g. Packet Inter arrival Time Mean, Packet count, Byte Size)에 기반한 분석 방법에 비해 상대적으로 빠른 분석이 가능하다. 시그니쳐 추출 작업은 반자동화로 이루어 진다. User-Agent, 도메인 필드에 대한 시그니쳐는 추출 시스템에 의해 추출되고, 호스트 필드는 시그니쳐 추출 시스템에 의해서 추출되지만 시그니쳐로 사용 여부는 관리자가 판단한다. Uri필드는 추출 시스템에서 전체 uri가 추출되며 시그니쳐로 사용 가능한 문자열은 관리자가 추출한다. 호스트 필드의 포트 번호와 User-Agent 필드의 버전 정보는 제거한 후, 해당 필드 값으로 추출된다.

추출된 시그니쳐는 제공되는 서비스의 종류에 비해 상대적으로 클라이언트 프로그램 종류, 기능의 개수는 적기 때문에 프로그램의 종류와 기능은 중복적으로 나타난다. 중복된 부분의 필드를 분류 시스템에서 반복하여 매칭하는 것을 불필요한 과정이다. 따라서 본 논문에서는 계층 구조 기반의 시그니쳐 트리를 제안한다. 계층 구조 기반의 시그니쳐 트리는 동일한 필드를 트리의 노드로 구성하여 반복 매칭을 방지할 수 있다. 또한 트리의 부모 노드에 포함된 시그니쳐만을 탐색하기 때문에 분석 시스템의 시그니쳐 탐색 공간을 최소화할 수 있다. 그림 3은 시그니쳐 트리 구조의 예시를 나타내고 있다.

루트를 제외한 트리의 깊이는 4 레벨로 구성된다.

Request packet
GET /checkupdate.php?cl=&v=103246420 HTTP/1.1 Host : update.utorrent.com:7070 User-Agent : BTWebClient/3130(27220) AcceptEncoding : gzip Connection : Close

field	Extraction method
User-Agent	User-Agent 키워드 다음부터 '/', \n' 전까지 (버전 정보는 시그니쳐에서 제외)
Domain	Host 키워드 다음부터':','\n'전까지 파싱한 후 com과 같은 1레벨 도메인 바로 앞에 오는 키워드를 도메인이라 정의 (포트번호, IP 제외)
Host	도메인 바로 앞에 오는 키워드를 호스트로 정의
Uri	Method필드 다음부터 HTTP 버전 전까지를 Uri로 정의 (시그니쳐가 될만한 키워드를 관리자가 직접 추출)

Result	
User-Agent = BTWebClient	Domain = utorrent
Host = update	Uri = checkupdate

그림 2. 필드 별 추출 방법
Fig. 2. Extraction method for each field

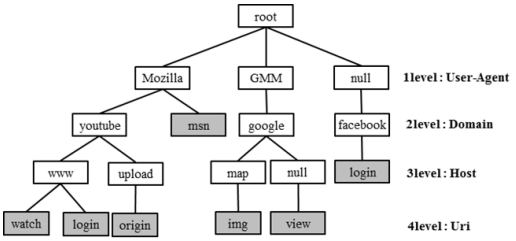

그림 3. 시그니쳐 트리 구조
Fig. 3. Signature tree structure

1-level User-Agent, 2-level 도메인, 3-level 호슽, 4-level Uri로 구성된다. 레벨을 정의하는 기준은 시그니쳐의 포함 관계에 의해서 규정된다. 1-level의 User-Agent를 구성하는 시그니쳐는 대부분 브라우져(IE, Chrome, firefox)를 나타내기 때문에 서비스의 종류에 비해 상대적으로 개수가 적다. 또한 1개의 브라우져를 통해서 다양한 서비스가 제공되기 때문에 여러 개의 서비스 시그니쳐를 포함하고 있다. 2-level의 도메인 필드는 서버에서 제공하는 서비스의 이름을 추출하는 필드로 1개의 도메인은 여러 개의 호스트를 포함하고 있다. 3-level은 호스트 필드로서 도메인이 포함하고 있는 여러 기능을 나타낸다. 마지막으로 4-level uri 시그니쳐는 1개의 호스트에서 제공하는 기능에 대한 정보를 나타내기 때문에 해당 호스트에서 제공하는 기능에 대한 Uri필드로 정의한다.

시그니쳐 트리 구조는 서비스 별로 분류가 가능하며 중복된 필드에 대한 반복 매칭을 방지할 수 있다. 또한 각각의 레벨은 포함 관계를 통해서 정의되기 때문에 시그니쳐의 탐색 공간을 줄일 수 있는 장점을 갖는다.

3.2 해시 테이블 기반 처리 속도 향상 방법

본 절에서는 계층 구조 기반의 HTTP 트래픽 분석 방법론의 분류 속도를 향상 시킬 수 있는 구현 방법에 대하여 기술한다.

계층 구조 기반의 시그니쳐 구조를 통해서 탐색 공간을 줄일 수 있지만, 자식 노드로 연결된 시그니쳐는 순차적으로 탐색해야하기 때문에 자식 노드가 많은 경우에는 탐색 공간이 증가하는 문제점이 있다. 따라서 본 논문에서는 트리의 각 레벨을 해시 테이블로 구성하고 해시 키 기반으로 시그니쳐의 탐색 공간을 최소화한다. 그림 4는 제안하는 HTTP 트래픽 분류 시스템의 전체 구성도를 나타내고 있다.

분석 시스템은 HTTP 트래픽과 필드 단위로 기술된 시그니쳐 리스트를 입력으로 받아서 3가지 분류

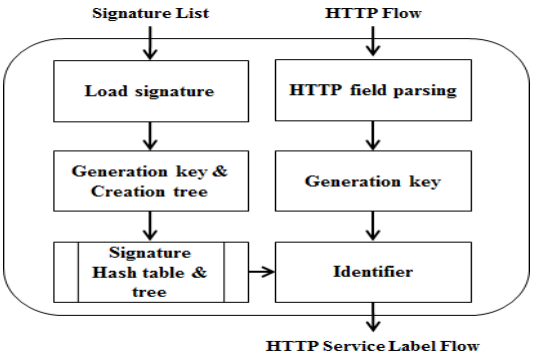

그림 4. 전체 구성도
Fig. 4. Overall architecture

기준이 식별된 플로우를 결과 값으로 도출한다. 입력된 HTTP 플로우의 페이로드는 HTTP 필드 단위로 파싱되며, 해시 함수를 통해 키 값으로 변환되어 분류 모듈의 입력으로 제공된다. HTTP 시그니쳐 리스트의 각 필드는 키 값으로 변환되고 트리 형태로 메모리에 적재된다.

본 논문에서 사용하는 해시 함수는 ELF(Executable and Linkable Format) 함수와 BJ(Bob Jenkins) 함수이다.[15,16] 두 함수를 모두 사용해서 해시 구조를 만들고 해시 테이블의 크기와 충돌 빈도를 고려하여 스트링 매칭에 적합한 해시 함수를 적용한다.

그림 5는 해시 기반 트리 구조를 보여준다. 해시 기반 트리 구조는 총 3-level로 구성 되어 있다. 1-level은 User-Agent와 도메인으로 구성되며, User-Agent와 도메인 필드 스트링을 합하여 1개의 키 값을 생성한다. 2-level과 3-level은 각각 호스트와 Uri로 구성되어 있다. 2-level은 호스트 필드의 스트링과 변환된 키 값으로 구성된다. 3-level의 uri는 스트링 값으로만 구성된다.

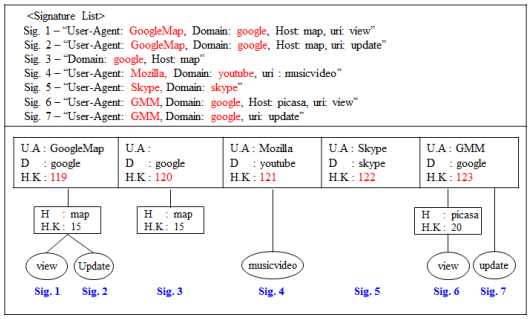

그림 5. 시그니쳐 해시 테이블 구조
Fig. 5. Signature hash table structure

트리 구조와 다른 점은 User-Agent, 도메인 레벨이 관리자의 개입이 없이 시그니쳐를 만들 수 있기 때문에 하나의 레벨로 구성 된다는 것이다. User-Agent는 세 가지 분류 기준 중에 응용만을 식별하기 위한 시그니쳐고, 도메인은 서비스만을 식별하기 위한 시그니쳐이다. 기능을 식별하기 위한 호스트 필드와 Uri 필드는 관리자의 개입이 필요한 시그니쳐이기 때문에 트래픽의 필드 값과 시그니쳐의 필드 값이 다를 수 있기 때문에 하나의 레벨로 구성되지 않는다.

다른 레벨과 다르게 Uri레벨은 키 값이 아닌 스트링 형태로 되어 있다. Uri 필드는 그림 2의 추출 과정에서도 알 수 있듯이, 필드 전체를 시그니쳐로 사용하지 않는다. 관리자의 판단에 의해 필요한 부분만 시그니쳐로 정의하기 때문에 트래픽의 Uri 필드와 키 매칭이 성립되지 않는다.

해시 기반 트리 구조와 HTTP 트래픽을 매칭하는 과정은 그림 6과 같다. 입력으로 들어온 하나의 플로우는 먼저 User-Agent와 도메인 레벨로 구성된 U-D 레벨 키 매칭을 수행한다. 키 매칭이 성공하면 *U-DLevelTextMatch()* 함수를 호출하여 텍스트 매칭을 수행하여 해시 키 매칭에 대한 검증 과정을 거친다. 해시 키를 생성하는 함수에서 다른 스트링이지만 같은 키를 생성할 수 도 있기 때문에 스트링 매칭을 통한 검증 과정이 요구된다. 텍스트 매칭까지 일치할 경우, 종료 플래그를 확인한다. 종료 플래그가 셋 되어 있으면 U-D 레벨에서 매칭을 종료 할 수도 있다. 하지만 더 자세한 분석을 위해 호스트 레벨과 Uri레벨의 검사를 진행 하고 실패할 경우, U-D레벨의 시그니쳐로 분석이 된다. 호스트 레벨에서의 검사는 U-D레벨과 마찬가지로 키 매칭을 우선적으로 하고, 매칭이 성공할 경우 텍스트 매칭을 수행한다. 텍스트 매칭이 성공 하면 종료 플래그를 확인하고 Uri 레벨 매칭을 수행한다. 호스트 레벨에 종료 플래그가 셋 되어 있고, Uri 매칭이 실패할 경우 호스트 레벨로 분석이 되고, Uri 레벨 매칭이 성공할 경우에는 Uri 레벨로 분석이 완료 된다. 2레벨인 호스트 레벨이 없을 경우에는 Uri 레벨로 매칭이 수행되며, 호스트 레벨 다음에 Uri 레벨이 없을 경우에는 호스트 레벨에서 매칭은 종료 된다.

```
1:  SLHT : Service Level Hash Table
2:  HLHT : Host Level Hash Table
3:  UL_SLHT : Sub uri list of SLHT node
4:  UL_HLHT : Sub uri list of HLHT node
5:
6:  for unlabeled each flow do
7:      if U-DLevelKeyMatch(flow, SLHT)
8:          if U-DLevelTextMatch(flow, sig)
9:              if sig.TermFlag setSigCode()
10:         else    return unknown
11:
12:         if HostLevelKeyMatch(flow, HLHT)
13:             if HostLevelTextMatch(flow, sig)
14:                 if sig.TermFlag setSigCode()
15:                 if UriLevelTextMatch(flow, UL_HLHT)
16:                     setSigCode()
17:             else
18:                 if UriLevelTextMatch(flow, UL_SLHT)
19:                     setSigCode()
20:                 else
21:                     if UriLevelTextMatch(flow, UL_SLHT)
22:                         setSigCode()
23:             return sigcode
24:
25:         else   return unknown
26:     done
```

그림 6. 매칭 알고리즘 슈더 코드
Fig. 6. Pseudocode of matching algorithm

Ⅳ. 성능 평가

본 장에서는 3장에서 제안한 방법론을 학내망의 실제 트래픽에 적용하여 분류 기준 별 시그니쳐 통계를 확인 하고, 서비스 기준에서의 분석 결과를 보여 준다. 또한 해시기반 트리 구조에서 사용한 두 가지 해시 함수의 성능 차이를 비교 하고, 마지막으로 Aho-Corasick 기반 분류 방법과 처리 속도 차이를 비교하여 그 타당성을 증명한다. 실험은 Intel Core i7 3.40GHz CPU, 8GB 메모리, 리눅스 커널 2.6.32 환경에서 수행되었다.

표 2는 실험에 사용된 트래픽 트래이스와 실험 기간을 보여주고 있다. 학내 망과 인터넷의 연결 지점에서 하루 동안 3,000여대의 호스트에서 발생한 트래픽을 페이로드 데이터를 포함한 플로우 형태로 수집하였다. 전체 트래픽 중에 HTTP 트래픽은 플로우 기준으로 총 25.8%가 발생 되었고, 패킷 기준으로는 42.2%, 바이트 기준으로는 48.6%가 발생 되었다. 바이트 기준으로 약 50%가 HTTP 트래픽이고 점점 증가하는 추세를 보이고 있기 때문에, HTTP 트래픽 분석이 필요하다.

표 2.트래픽 트레이스
Table 2. Traffic trace

	Flow	Packet	Byte	Duration
Total	41,365K	2,712M	2,263G	1day
HTTP	10,672K	1,144M	1,088G	1day

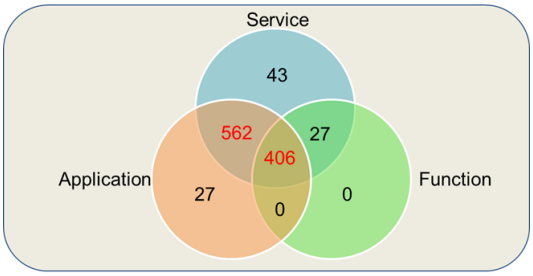

그림 7. 시그니쳐 분포
Fig. 7. Signature distribution

그림 7은 HTTP 트래픽을 분석하기 위해 추출한 시그니쳐 통계 정보를 보여준다. 총 HTTP 시그니쳐는 1,065개이고 대부분의 시그니쳐가 서비스와 응용 기준의 분류가 가능하다.

표 3은 1,065개의 시그니쳐로 HTTP 트래픽을 분석한 분석 결과이다. 추출된 1,065개의 시그니쳐로 HTTP 트래픽만을 분석 했을 때, 분석 결과는 플로우 단위로 66.2%, 패킷 단위로 74.7%, 바이트 단위로 78.6%를 분석 할 수 있었다.

표 4는 분석된 결과를 서비스 기준으로 재분석 하였을 때 서비스 순위를 나타낸다. 학내 망에서 가장 많은 트래픽을 발생 시킨 서비스는 Naver라는 다양한 기능을 제공하는 포탈 서비스 이다. 분석된 HTTP 트래픽을 기준으로 했을 때, 플로우 단위로 37.2%, 패킷 단위로 26.1%, 바이트 단위로 25.8%를 발생 시켰으며, 상위권에 위치한 대부분의 서비스들은 Naver와 같은 포탈 사이트들이라는 것을 확인 할 수 있다.

해시 함수의 성능을 평가하기 위해서 User-agent와 도메인 시그니쳐를 해시 키로 생성하여 해시 크기에 따른 충돌 빈도를 측정하였다. 그림 8은 해시 사이즈에 따른 충돌 빈도를 비교한 그래프이다.

두 해시 함수 중에 BJ가 해시 사이즈가 적을수록 충돌이 적게 난다는 것을 알 수 있다. 해시 사이즈가 커지면 두 함수의 차이는 거의 없다고 볼 수 있기 때문에, 해시 사이즈가 적을 때 충돌이 적은 BJ 함수가 성능이 더 우수하다고 판단 할 수 있다. 또한 두 해시 함수로 처리 속도 또한 비교를 해 보았지만 처리 속도

표 3. 분류 결과
Table 3. Classification result

	Flow	Packet	Byte
Total HTTP	10,672K (100%)	1,144M (100%)	1,088G (100%)
Classified HTTP	7,064K (66.2%)	855M (74.7%)	864G (78.6%)

표 4. 상위 10개 서비스
Table 4. Top 10 services

Rank	Service	Flow	Packet	Byte
1	Naver	1,482K (37.2%)	52,470K (26.3%)	43,172M (25.8%)
2	Korea	649K (16.3%)	43,954K (22.1%)	31,760M (18.9%)
3	Daum	470 (11.8%)	19,134K (9.6%)	16,540M (9.9%)
4	Nate	158K (3.9%)	11,163K (5.6%)	11,319M (6.8%)
5	Google	69K (1.7%)	13,435K (6.7%)	10,420M (6.2%)
6	Dropbox	62K (1.5%)	6,458K (3.2%)	5,523M (3.3%)
7	Microsoft	40K (1.1%)	3,415K (1.7%)	3,248M (1.9%)
8	Gmarket	35K (1.8%)	2,224K (1.1%)	2,222M (1.3%)
9	Me2day	32K (0.8%)	2,024K (1.0%)	2,104M (1.2%)
10	youtube	27K (0.7%)	1,913K (0.9%)	1,821M (1.1%)

에서 큰 차이를 보이지 않았기 때문에 본 논문에서는 충돌 횟수가 더 적은 BJ 해시 함수를 사용 한다.

시그니쳐 개수가 증가하면 트래픽 분류 시스템의 처리 속도가 느려 질 수 있다. 하지만 그림 9와 같이 해시 구조를 사용한다면 시그니쳐의 수가 아무리 증가 한다고 해도 속도 차이에는 큰 변함이 없다는 것을 알 수 있다.

제안하는 방법의 성능 평가를 위해서 Snort에서 적용하고 있는 Aho-Corasick 패턴 매칭 알고리즘과 처리 속도를 비교하였다. 두 방법은 동일한 전체 시그니쳐를 하나의 시그니쳐 트리로 구성하여 매칭을 수행한다. 그림 10은 해시 기반 트리 구조와 Aho-Corasick 기반 분석의 처리 속도를 비교한 그래프이다.

학내 망 트래픽 특성상 19시~09시까지는 트래픽의

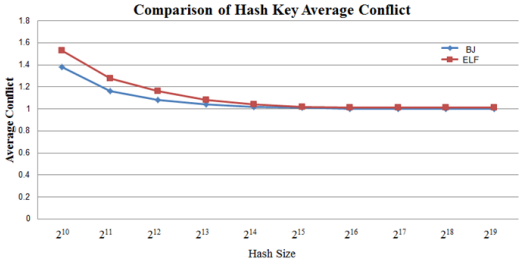

그림 8. 해시 테이블 크기에 따른 충돌 비교
Fig. 8. Comparison of hash key conflict according to hash size

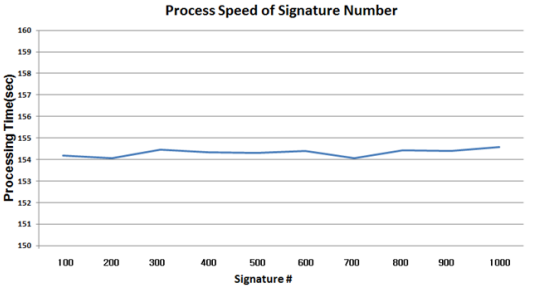

그림 9. 시그니쳐 개수에 따른 처리 시간 비교
Fig. 9. Comparison of processin time according to the number of signatures

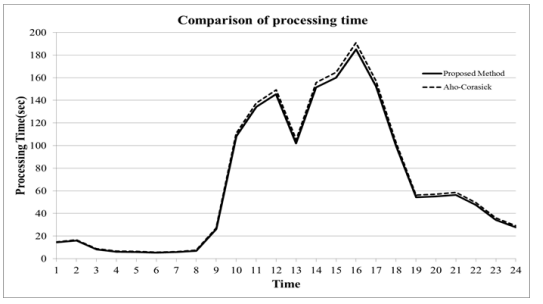

그림 10. AC와 제안하는 방법의 처리 시간 비교
Fig. 10. Comparison of processing time AC vs. proposed method

양이 많지 않으며, 일과 시간 동안에는 200Mbps 이상의 트래픽이 발생한다. Aho-Corasick과 해시 기반 트리 구조와의 처리 속도는 트래픽 양이 적을 때는 크게 차이가 나지 않는다. 하지만 트래픽이 양이 많을 때는 Aho-Corasick과 차이가 나타나는 것을 확인 할 수 있다. 1주일간의 트래픽을 대상으로 동일한 실험을 한 결과, 월-금요일까지는 동일한 패턴으로 나타났으며, 주말에는 트래픽 양이 많이 않아서 AC와 큰 성능 차이를 보이지 않았다. 표 5는 Aho-Corasick과 제안하는 방법의 트래픽 분석 시간 최소값, 최대값, 평균값 차이를 보여준다. 최소 0.4초, 최대 5.5초 정도의 차이를 보이며, 평균적으로는 2초 정도 차이가 발생 한다.

표 5. 최소, 최대, 평균 처리 시간
Table 5. Min, max, average processing time

	Min(sec)	Max(sec)	Avg.(sec)
Aho-Corasick	5.77	190.77	69.17
Proposed method	5.34	185.02	67.07

V. 결론 및 향후 과제

본 논문에서는 HTTP 트래픽의 서비스 별 분석을 위해서 시그니쳐 계층 구조를 정의하고, 이를 기반으로 분류 시스템의 처리 속도를 향상 시킬 수 있는 방법을 제안하였다. 제안하는 분류 방법론은 Aho-Corasick 기반 분석 방법과 비교해 한 시간 트래픽을 분석하는데 최대 5초 이상의 처리 속도를 줄일 수 있었고, 평균적으로 2초 이상의 처리 속도를 감소 시킬 수 있었다.

본 논문에서는 HTTP 트래픽의 분석 시스템의 처리 속도를 향상 시키기 위해서 소프트웨어적인 해결 방안을 제시하였다. 제안하는 방법을 기반으로 하드웨어를 접목하여 고속 링크의 대용량 트래픽을 실시간으로 분석할 수 있는 프레임워크를 구성하는 연구를 수행할 계획이다.

References

[1] W. Li, A. W. Moore, and M. Canini, "Classifying HTTP traffic in the new age," in *Proc. ACM SIGCOMM*, pp. 479-489, Washington, USA, Aug. 2008.

[2] Y. Bhole and A. Popescu, "Measurement and analysis of HTTP traffic," *J. Network and Systems Management*, vol. 13, no. 6, pp. 357-371, Dec. 2005.

[3] J. S. Park, S. H. Yoon, and M. S. Kim, "Performance improvement of the payload signature based traffic classification system using application traffic locality," *J. KICS*, vol. 38B, no. 7, pp. 519-525, Jul. 2013.

[4] S. H. Yoon, H. G. Roh, and M. S. Kim, "Internet application traffic classification using traffic measurement agent," in *Proc. KICS*, pp. 1747-1750, Jeju Island, Korea, Jul. 2008.

[5] C. S. Park, J. S. Park, and M. S. Kim, "Study on automatic payload signature generation system using protocol filter," in *Proc. KICS*, pp. 655-656, Jeju Island, Korea, Jun. 2013.

[6] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, "Lightweight, payload-based traffic classification an experimental evaluation," in *Proc. IEEE Commun.*, pp. 5869-5875, Beijing, China, May 2008.

[7] J. H. Kim, S. H. Yoon, and M. S. Kim,

"Study on traffic classification taxonomy for multiliteral and hierarchical traffic classification," in *Proc. APNOMS*, pp. 1-4, Seoul, Korea, Sept. 2012.

[8] J. H. Kim, S. H. Yoon, and M. S. Kim, "Research on traffic taxonomy for internet traffic classification," in *Proc. APNOMS*, Taipei, Taiwan, Sept. 2011.

[9] J.L. Garcia-Dorado, J.A. Hernandez, J. Aracil, J.E.L. Vergara, F.J. Montserrat, E. Robles, and T.P. Miguel, "On the duration and spatial characteristics of Internet traffic measurement experiments," *IEEE Commun. Mag.*, vol. 46, no. 11, pp. 148-155, Nov. 2008.

[10] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 Machine Learning Algorithm," in *Proc. ICNC*, pp. 244-248, Maui, HI, USA, Feb. 2012.

[11] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis, "Regular expression matching on graphics hardware for intrusion detection," in *Proc. RAID*, pp. 265-283, Saint-Malo, France, Sept. 2009.

[12] M. Roesch. "Snort – lightweight intrusion detection for networks," in *Proc. USENIX LISA*, pp. 229-238, Washington, USA, Nov. 1999.

[13] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333-340, Jun. 1975.

[14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, and C. Stein, *Introduction to Algorithms* 3rd Ed., The MIT press, 2009.

[15] ITS Committee, Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2, May 1995.

[16] B. Jenkins, "A new hash functions for hash table lookup," *J. Dr. Dobb's*, Sept. 1997.

최 지 혁 (Ji-Hyeok Choi)

2012년 : 고려대학교 컴퓨터정보학과 졸업
2014년 : 고려대학교 컴퓨터정보학과 석사 졸업
2014년~현재 : 한국전자통신연구소 연구원

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 트래픽 분류

박 준 상 (Jun-Sang Park)

2008년 : 고려대학교 컴퓨터 정보학과 졸업
2010년 : 고려대학교 컴퓨터 정보학과 석사
2010년~현재 : 고려대학교 컴퓨터 정보학과 박사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 트래픽 분류

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산학과 졸업
2000년 : 포항공과대학교 컴퓨터 공학과 석사
2004년 : 포항공과대학교 컴퓨터 공학과 박사
2006년 : Post-Doc. Dept. of ECE, Univ. of Toronto, Canada
2006년~현재 : 고려대학교 컴퓨터정보학과 부교수

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크