목 차

그림 목차

표 목차

1 서론

인터넷이 발전하면서 멀티미디어 서비스와 대용량 데이터 전송 등 다양한 서비스를 제공하기 위한 애플리케이션이 등장하고, 웜 바이러스와 같이 네트워크에 문제를 일으키는 트래픽이 등장하고 있다. 네트워크 관리자는 다양한 애플리케이션에서 발생하는 네트워크 트래픽을 원활히 지원할 수 있도록 네트워크의 Capacity을 계획하기 위해서, 또한 웜 바이러스 같이 네트워크와 서비스에 문제를 야기하는 트래픽을 구분하기 위해서 각 애플리케이션 별 네트워크 트래픽을 분류하는 것은 매우 중요하다. 특히 적합한 QoS(Quality of Service)를 제공하고 안전한 네트워크 환경을 만들기 위해서 정확한 애플리케이션 트래픽 분류는 필수적이다.

현재 애플리케이션 트래픽 분류는 크게 다음 두 가지 종류의 접근 방법을 사용한다. 첫째는, IANA[20]에서 할당한 Port 번호를 기반으로 하는 분류로 애플리케이션에서 사용하는 Well-known Port 번호를 분석하는 방법이다. 다음은 애플리케이션의 Payload 분석을 통해 특정 Signature를 추출하고 이를 애플리케이션 분류에 적용하는 방법이다.

하지만, 최근에 동적으로 Port 번호를 할당하여 Packet을 발생하거나 Payload를 암호화한 Packet을 발생하는 애플리케이션이 많아지고 있어서 기존의 방법을 통한 애플리케이션 트래픽 분류의 정확도가 떨어지고 있다. 게다가 IANA에서 할당한 기존의 Well-known Port 번호를 다른 목적으로 이용하는 애플리케이션이 증가하고 있다. 또한 Payload 분석을 통한 트래픽 분류는 Packet의 Header 뿐만 아니라 Payload까지 저장해야 하므로 많은 데이터 저장 공간이 필요하고, Payload도 분석해야 함으로 분류를 수행하기 위해 많은 시간이 요구되는 문제점을 가지게 된다.

이에 따라 애플리케이션의 동적인 Port 번호 사용과 암호화된

Packet 발생에 대처하는 방안으로 Packet Header 정보들의 통계 정보를 이용한 애플리케이션 트래픽 분류 방법[23-26]과 Packet Header의 통계 정보를 ML(Machine Learning) 알고리즘에 적용한 애플리케이션 트래픽 분류 방법이 등장했다. ML 알고리즘을 이용하여 동적으로 변하는 Port 번호의 패턴을 찾아낼 수 있다는 점과 Payload에 독립적인 Inter-packet Arrival Time, Packet Size Statistics, Byte Count 등의 통계수치를 이용한다는 점은 ML 알고리즘이 애플리케이션의 변화에 대처할 수 있음을 보여준다. 현재 ML 알고리즘을 이용한 애플리케이션 트래픽 분류에 대한 많은 연구가 진행되고 있다[2-9, 15, 22]. 하지만, 기존 연구들이 수행한 실험 방법은 실제 Internet Link의 트래픽을 분석하기에 세가지 관점에서 적합하지 않다.

첫째, 기존 연구들[2-9, 15, 22]은 ML 알고리즘을 이용하여 애플리케이션 트래픽을 분류하는 것이 매우 정확한 결과를 가진다는 점을 보여주고 있다. 그러나 ML 알고리즘을 적용한 애플리케이션 트래픽 분류를 수행하는 기존 연구들은 Well-known Port 번호를 사용하거나 또는 그 외에 다른 애플리케이션에서 사용하지 않는 고정된 Port 번호를 사용하는 애플리케이션만을 선택하여 그 해당 애플리케이션의 데이터를 수집하여서 분류한 결과를 보여주고 있다. 즉, 기존 연구에서 보여주는 애플리케이션 트래픽 분류 결과는 현재 변화하는 애플리케이션의 특성을 잘 반영하지 못한다는 단점을 가지고 있다. 이를 보완하기 위해서 본 논문에서는 Port 번호를 기반으로 애플리케이션을 분석하는 것이 아니라 각 애플리케이션이 발생하는 트래픽을 실제로 수집하여 분석하는데 사용하였다. 즉, Dynamic하게 변하는 Port 번호를 사용하거나 웹 트래픽을 위한 Port 번호인 80번과 같은 Well-known Port 번호를 사용하는 Web Disk나 P2P 애플리케이션을 직접 모니터링하여 애플리케이션 트래픽 분류를 수행하며 그 결과를 제시한다.

또한, 기존 연구에서 제시된 트래픽 분류 결과의 정확도를 평가

함에 있어 대부분의 연구는 동일한 시간대에 수집된 하나의 Closed 데이터 Set 안에서 Training Set과 Testing Set을 구성하는 Cross Validation을 채택하고 있지만, 이는 네트워크 트래픽을 모니터링하고 분석하는 실제 네트워크 운영자 입장에서는 현실적으로 적용하기 어렵다. 이를 보완하기 위해서 ML 알고리즘을 적용하여 Training을 시키는 Training Set과 정확도에 사용되는 Testing Set을 분리하는 Split Validation 기법이 적용되어야 한다.

마지막으로, ML 알고리즘을 이용한 애플리케이션 트래픽 분류를 한 기존 논문들은 Flow를 기반으로 한 정확도를 제시하고 있다. 하지만 실제로 Byte를 기반으로 한 정확도가 Traffic Shaping, Usage Billing Policy, Network Planning과 같은 분야에 실제적으로 적용하기에 더 유용하다[11]. 또한 Flow를 기반으로 한 정확도는 각 Flow마다 Size가 다른데 이것을 모두 같은 한 개의 Flow로 보고 분석하므로 각 Flow가 가지는 가중치를 제대로 반영하지 못한다. 본 논문에서는 이러한 점을 반영하여 Flow 기반 정확도뿐만 아니라 Byte 기반의 데이터를 이용하여 애플리케이션 트래픽 분류를 수행하고 그 결과를 비교 분석한다.

본 논문에서는 ML 알고리즘을 적용한 애플리케이션 트래픽 분류의 기존 연구에서 찾을 수 있는 문제점을 자세하게 설명하고 그 문제점을 해결할 수 있는 방안을 제시한다. 그리고 제시한 방안에 기반하여 POSTECH의 인터넷 트래픽을 대상으로 J48, REPTree, BayesNet, Multi-layer Perceptron의 4가지 대표적인 ML 알고리즘을 이용한다. 그리고 두 개 이상의 동적인 Port 번호를 사용하거나 또는 기본 Port 번호로 80번을 사용하는 Web Disk와 P2P 애플리케이션을 포함한 POSTECH의 구성원들이 많이 사용하는 애플리케이션을 대상으로 애플리케이션 트래픽을 분류하고 그 결과를 분석한다. 이러한 결과를 기반으로 Overall Accuracy 관점에서 애플리케이션 트래픽 분류를 위한

ML 알고리즘과 Feature Set을 실험을 통하여 제시하고자 한다.

마지막으로 POSTECH의 인터넷 트래픽을 대상으로 ML 알고리즘을 적용한 애플리케이션 트래픽 분류를 수행한다. 이 애플리케이션 트래픽 분류에 사용하는 트래픽은 분류의 정확도를 알기 위해서 직접 발생시킨 트래픽을 사용하였다. 또한 기존의 Approach를 기반으로 한 NGMON[13]을 적용한 애플리케이션 트래픽 분류를 수행한다. 이 두 가지 Approach를 적용한 결과를 비교 및 분석함으로써 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 기존의 Approach를 적용한 결과와 다른 점을 제시하고 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 좀 더 높은 정확도를 가질 수 있도록 나아가야 할 방향을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 애플리케이션 트래픽 분류에 적용되는 ML 알고리즘에 대해서 살펴보고, ML 알고리즘을 적용하여 애플리케이션 트래픽 분류를 수행한 기존 연구에 대해서 알아본다. 그리고 기존 연구를 Internet Link의 네트워크 트래픽 분류에 적용하기 위해서 고려해야 할 점에 대해서 논의하고 그것을 해결하기 위한 방안을 제안한다. 3장에서는 앞 장에서 제시한 기존 연구의 문제점의 해결책을 기반으로 한, 애플리케이션 트래픽 분류에 적용할 수 있는 방안을 제시하며, 여러 ML 알고리즘과 Feature Set을 적용한 애플리케이션 트래픽 분류 결과를 제시한다. 4장에서는 ML 알고리즘과 NGMON을 적용한 애플리케이션 트래픽 분류 결과를 비교 분석한다. 마지막으로 5장에서는 결론 및 향후 연구를 제시한다.

2 관련 연구

이 장에서는 애플리케이션 트래픽 분류에 적용되는 ML(Machine Learning) 알고리즘에 대해서 간략하게 살펴보고, ML 알고리즘을 이용한 애플리케이션 트래픽 분류의 기존 연구에 대해 정리한다. 그리고 본 논문에서 사용한 애플리케이션 트래픽 분류의 정확도 평가 방법에 대해서 설명한다. 마지막으로 ML 알고리즘을 적용한 애플리케이션 트래픽 분류의 기존 연구 방법을 실제 Internet Link상의 애플리케이션 트래픽 분류에 적용할 때 고려해야 할 점을 정리한다

2.1 ML 알고리즘

애플리케이션 트래픽 분류에 사용하는 ML 알고리즘은 Unsupervised, Supervised, Feature Reduction, Neural Network 알고리즘으로 나누어서 생각해 볼 수 있다. [2, 3]에서 사용하는 Unsupervised ML 알고리즘은 주어진 데이터들의 유사성을 기반으로 각 클래스 별로 분류를 하며, 자주 사용하는 알고리즘으로는 K-Means[31], DBSCAN[28], AutoClass[32], Expectation Maximization[33] 등이 있다.

Expectation Maximization(EM) 알고리즘은 부분적으로 관측된 데이터를 가지고 모델링을 하기 위한 것이다. 주어진 통계적인 모델에서 D가 관측이 된 데이터 Set이고 U는 관측이 안된 데이터 Set이라고 하자. 파라미터 T는 이 두 데이터 Set을 표현하는 모델을 기술한다고 할 때 EM 알고리즘은 P(D|T) = sum{U} P(D, U|T)를 최대값을 갖도록 하는 T를 예측한다. 알고리즘은 초기 추정치 T를 이용하여서, 아래의 두 단계를 거쳐서 다음 단계의 파라미터 값을 예측한다.

- E-Step: 이 단계에서는 주어진 파라미터 Set과 관측 데이터를 이용하여서 관측이 안된 데이터에 대한 Posterior Probability 분

포 P(U|D, T)를 예측한다.

- M-Step: 이 단계에서는 E-Step에서 얻어진 U에 대한 Posterior 분포를 이용하여 전체 데이터 Set(관측된 것과 안된 것)에 대한 Log-likelihood를 최대화시키는 모델 파라미터 T를 얻는다.

$$Q(T' \mid D, T) = \text{sum}\{U\}\ P(D, U|T) * \log(P(D, U|T))$$

종종 E-Step은 전체 데이터를 표현할 수 있는 통계적인 파라미터(평균, 분산 등..)를 계산하는 과정으로 단순화 된다. 즉, 미리 전제된 통계 모델을 기술하는 파라미터를 관측이 된 데이터에 기반하여서 예측한다.

K-Means 알고리즘은 Gaussian 분포를 모델로 한 EM 알고리즘으로 볼 수 있다. 주어진 데이터가 여러 개의 Gaussian 분포의 혼합으로 여기고 각각의 Gaussian 분포를 기술하는 파라미터를 구하는 작업이 된다.

DBSCAN 알고리즘은 Noise가 존재하는 데이터의 Density-based Spatial Clustering 이다. Density-based 개념에 의거한 Cluster로서 한 개의 Cluster는 Density-connected로 된 점들의 최대 조로 이루어진다. Noise가 있는 임의 외형의 공간 데이터 Bases에서 Cluster를 찾는다 [29].

Nearest Neighbor(NN) Classification은 분류하고자 하는 클래스의 종류에 대해서는 알고 있지만 샘플들 각각에 대한 확률밀도함수(Probability Density Function)를 알지 못하는 상태에서 사용한다. 굳이 각 샘플에 대한 확률 파라미터들을 구하지 않고 샘플의 값을 그대로 좌표에 표시하여 Reference Set에 가장 유사하거나 거리 상으로 가까운 Group에 속하는 것으로 분류하는 방법이다.

AutoClass 알고리즘은 Maximum Posterior Probability Classification을 찾아내는 Unsupervised Bayesian Classification이다. 이 알고리즘은 Group

의 수를 자동적으로 결정해 주고, Discrete와 Real Valued 데이터를 혼합해서 사용하는 것을 허용하며, Missing Value를 처리할 수 있다. 많은 양의 데이터를 Linear한 시간에 처리할 수 있다. 하나의 Group내에서 Feature들 사이의 Correlation을 허용한다. 각 Group간의 연간관계의 Probabilistic을 알아보며, 분류 되는 각 Group에 대한 Description을 제공한다[30].

이론적으로 Unsupervised ML 알고리즘을 사용하면 사람이 미쳐 파악하지 못한 애플리케이션까지 그룹으로 묶어줘 주어진 데이터를 분류할 수 있다는 장점이 있다. 하지만 주어진 데이터가 정확히 어떤 애플리케이션인지 알 수 없다. 그룹 지어진 데이터를 몇 개의 애플리케이션으로 구분을 해야 하는가의 문제는 ML 알고리즘분야에서 Open Problem으로 남겨져 있으며, 따라서 미리 구분이 되어야 하는 애플리케이션의 개수를 정해 주어야 한다.

[4, 5, 6]에서 사용하는 Supervised ML 알고리즘은 알려진 데이터를 이용하여 모델을 Training 한 후, 그것을 기반으로 알려지지 않은 데이터를 Testing 과정을 거쳐서 분류하는 알고리즘을 말하며 자주 사용되는 알고리즘으로는 J48[34], REPTree[35], NaïveBayes[36], BayesNet[37], Naïve Bayes Kernel Estimator[38]등이 있다.

Naïve Bayes는 임의의 데이터가 특정 분류에 속할 확률을 계산하여 계산된 확률 중 가장 확률이 높은 분류를 선택하는 것을 말한다. 즉, $P(a_i/b_i) = P(a_i)P(b_i \mid a_i)/P(b_i)$ 와 같은 Bayesian 규칙으로 확률을 계산한다. 이 식의 의미는 $b_1, b_2, \ldots, b_k$중에서 b_i가 Group $a_i \sim a_m$에 속할 확률을 의미하고, 각각의 결과값 중 가장 큰 값을 선택합니다.

Naïve Bayes Tree는 Naïve Bayes와 Decision Tree Classifier와의 Hybrid이다. Naïve Bayes Classifier가 Leaf Node인 것이 Branch와 Leaf로 구성된 Decision Tree Classifier이다. 이것은 많은 Training 데이터를 적용하는 것이 가능하다.

Bayesian Network는 변수를 표현하는 Node와 변수들 간의 의존관계를 표현하는 호(arc)의 방향성 비 순환 그래프(Directed Acyclic Graph)이다. Node A에서 Node B까지의 호가 있다면 A는 B의 Parent라고 부른다. Node가 값이 주어져 있다면 Evidence Node라고 부른다 하나의 Node는 측정값, 인수, 숨겨진(Latent) 변수, 가설 등의 어떤 종류의 변수일 수도 있다. Node는 임의의 변수를 표현하는데 제약이 없다. 이것이 Bayesian Network에 대해서 "Bayesian"이라는 것이다. Bayesian Network은 그래프(Graph) 상에 Node에 의해 표현되는 모든 변수에 대한 Joint Distribution의 표현이다. 변수를 X(1),…,X(n), Parents(A)는 Node A의 Parents라고 하자. 이때 X(n)를 통한 X(1) 의 Joint Distribution 은, I 값이 1부터 n까지 일 때, 확률분포 P(X(i) | Parents(X(i)))의 곱으로서 표현된다. 만일 X가 Parents를 갖지 않을 경우는 그것의 확률분포는 Unconditional이라고 하며, Parents가 있을 경우는 Conditional이라고 한다. 변수들간의 의존(Dependency)에 대한 의문들은 그래프만 연구하면 해결될 수 있다. D-separation이라는 그래프 표기는 조건부 독립(Conditional Independent)의 표기와 동등한 것이라고 할 수 있다. 만일 Node X와 Y가 D-separated되어 있다면 (명확하게 주어진 Evidence Node에 대해), 변수 X와 Y는 주어진 Evidence Variable에 대해 독립적(Independent)이다[40].

J48과 REPTree와 같은 Decision Tree는 과거에 수집된 데이터의 Record를 분석하여 이들 사이에 존재하는 패턴, 즉 부류 별 특성을 속성의 조합으로 나타내는 분류모형을 나무의 형태로 만드는 것이다. 그리고 이렇게 만들어진 분류 모형은 새로운 Record를 분류하고 해당 부류의 값을 예측하는 데 사용된다. 이것은 순환적 분할(Recursive Partitioning) 방식을 이용하여 나무를 구축하는 기법으로, 나무의 가장 상단에 위치하는 뿌리마디(Root Node), 속성의 분리기준을 포함하는 내부마디(Internal Nodes), 마디와 마디를 이어주는 가지(Link), 그리고 최종 분류를 의미하는 Leaf들로 구성된다[41].

Neural Network는 생물학에서의 Neural Network를 수학적이며 계산학적인 모델로 표현한 것이다. Neural Network는 인공적인 뉴런들의 상호 연결된 것들로 이루어져 있다. 대부분의 경우에 Artificial Neural Network는 학습 기간 동안에 네트워크를 통해서 흘러 다니는 외부 또는 내부 정보들을 기반으로 네트워크의 구조를 바꿔가며 적응하는 시스템이라 할 수 있다. Neural Network는 Input Node, Output Node, Hidden Node의 관계로 이루어져 있으며, 결론적으로 데이터의 패턴을 찾아내는 기능을 한다고 볼 수 있다. Input Node와 Output Node는 입력 값과 출력 값을 지정하는 곳으로 어떤 값을 집어 넣을지 어떤 값을 얻을지 결정하는 곳이다. Input Node에서 들어간 X값은 여러 Hidden Node들을 거치며 계산되어 Output Node로 Y값을 출력한다. 여기서 계산은 Sigmoidal 함수와 같은 정해진 특정함수를 이용한다. Neural Network에서 자주 사용하는 알고리즘으로는 Radial Base Function(RBF) Network와 Multi-layer Perceptron(MLP) 등이 있다[18]. 아직까지 Neural Network를 애플리케이션 트래픽 분류에 적용한 기존 연구는 찾아보기 어려우며 본 논문에서는 Neural Network를 적용한 애플리케이션 트래픽 분류의 결과도 제시하고자 한다.

Williams[15]은 Feature의 개수가 많을수록 분류의 Overall Accuracy가 높아짐을 보여준다. 하지만 Feature의 개수가 많을수록 계산 복잡도가 높아지는 즉, 수행시간이 오래 걸리는 단점이 있다. 이 Trade-off 문제를 최적화하여 해결하기 위해서 Feature Reduction 알고리즘을 사용한다. 그 예로서 Genetic 알고리즘[9], FCBF(Fast Correlation-based Feature Selection)[17], Consistency-based Search in Feature Selection[16] 등이 있다.

FCBF[17]는 최적의 Feature를 Selection 할 수 있는 Heuristic 기법 중 본 연구와 관계되어서 많이 쓰이고 있는 기법이다. 이 기법은 아래와 같이 Filter Model과 Wrapper Model, 두 개의 Step을 통해서 유용

한 Feature Set을 선택하게 된다.

- Filter Model: ML 알고리즘이 무엇이든지 관계없이, Training 데이터의 일반적인 특성에 의존하여 Feature들을 선택하는 모듈이다.
- Wrapper Model: Filter Model에서 선택된 Feature들 중 중복되지 않은 Feature들을 선별해 내고, 중복적인 것 중 하나를 고르게 될 경우, 어떤 것을 선택할 것인지에 대해서, 그 Feature들이 가진 성능 등을 평가하여서 판단하는 모듈이다. 이 경우, 어떤 ML 알고리즘을 선택하느냐에 따라서 선택하는 Feature가 달라지게 된다.

Genetic 알고리즘은 최적화 문제에서 기존의 다른 알고리즘보다 전역적이고 강인한 최적화 방법을 제공한다. 고전적인 최적화 알고리즘들은 그 시스템에 대한 충분한 지식과 수학적인 해석에 의해 설계되었음에도 불구하고 주어진 환경에 따라 국부 최소에 빠질 가능성이 있지만, 유전 알고리즘은 이에 비해 목적 함수에 대한 수학적 제약이 거의 없기 때문에 다양한 분야에의 적용이 가능하며 전역적 최적 해를 발견할 가능성이 높다. 유전 알고리즘이 기존의 다른 최적화 알고리즘과 구별되는 점들을 정리해보면 다음과 같다[39].

첫째, 유전적 알고리즘은 파라미터 자체가 아닌 파라미터 집합에 기반을 두어 탐색하는 방법이다. 둘째, 유전 알고리즘에서의 탐색은 결정론적 규칙이 아닌 확률에 기반을 둔 연산자를 사용하여 수행된다. 셋째, 탐색 공간에 대한 연속성이나 미분 가능성 등의 최적화 함수의 정보를 필요로 하지 않고 단지 적합도 함수 값만을 사용한다. 넷째, 유전 알고리즘은 탐색 공간에 대한 지식이 없고 특성을 잘 알지 못하는 경우에도 적용이 가능하며 알고리즘이 간결하고 강인하다. 다섯째, 병렬 처리가 가능하며 전역적인 최적 해를 구할 가능성이 다른 알고리즘에 비하여 크다[39].

이 Feature Reduction 알고리즘은 주어진 Feature들 중 중복된 속성을 가진 Feature들을 제외하고 Overall Accuracy를 높인다고 판단하기 힘든 Feature들을 제외한 나머지 Feature들로 구성한 Feature Set을 자동으로 정의한다.

애플리케이션 트래픽 분류의 새로운 Approach로서 ML 알고리즘과 다른 Approach를 적용했을 때의 차이점은 분류하고자 하는 애플리케이션을 제외한 나머지 애플리케이션을 'UNKNOWN으로 정의 할 수 있다 또는 없다'의 차이이다. ML 알고리즘을 적용한 애플리케이션 트래픽 분류에서는 모든 Testing Set의 데이터가 Training Set에 있는 애플리케이션으로 분류되기 때문에 UNKNOWN을 정의하기 어렵다. 반면에 다른 Approach를 적용한 애플리케이션 트래픽 분류에서는 UNKNOWN을 정의할 수 있다.

2.2 ML 알고리즘을 이용한 애플리케이션 트래픽 분류

이 장에서는 ML 알고리즘을 이용한 기존의 애플리케이션 트래픽 분류 연구에 대해서 살펴본다. ML 알고리즘을 이용한 애플리케이션 트래픽 분류 연구는 1990년대 말부터 이루어 졌으며 현재 CAIA(Center for Advanced Internet Architecture)[1] 등에서 많이 이루어지고 있다. CAIA는 장기적인 프로젝트로 A Study of Hidden Internet Traffic을 진행하고 있으며, 이를 통해서 ML 알고리즘을 적용한 애플리케이션 트래픽 분류 연구를 진행하고 있다. 기존 연구들은 크게 Unsupervised ML 알고리즘을 이용한 경우, Supervised ML 알고리즘을 이용한 경우, Unsupervised와 Supervised ML 알고리즘을 함께 이용한 경우, 그리고 Feature Reduction 방법을 이용한 경우 이렇게 4가지로 나누어 볼 수 있으며 이에 따라서 사용한 알고리즘도 다르다. 표 1은 기존 연구에서 사용한 알고리즘을 보여주고 있다.

ML 알고리즘	Erman et al. [2]	Zander et al. [3]	Nguyen et al. [4]	Park et al. [5]	Andrew et al. [6]	Erman et al. [7]	Williams et al[8]	Park et al. [9]
K-Means	O							
DBSCAN	O							
Autoclass	O							
Expectation Maximization		O				O		
Naïve Bayes			O	O	O	O	O	
NBKE				O	O			O
REPTree				O				O
J48								O
C4.5							O	
Bayesian Network							O	
Naïve Bayes Tree							O	
FCBF					O			O
GA								O

표 1. 기존 연구 논문들이 사용한 ML 알고리즘

기존 연구 논문들이 적용한 ML 알고리즘은 다 다르며, 각 논문들이 애플리케이션 트래픽 분류에 적합하다고 제안한 ML 알고리즘도 다르다. 또한 기존 연구들은 각각 다른 Feature들을 선택해서 사용하였다. 표 2는 기존 연구들이 사용한 Feature들을 보여주고 있다. 모든 기존 연구들이 공통적으로 Inter-packet Arrival Time을 선택한 것을 볼 수 있다. 기존 연구 논문들이 Feature로 Port 번호를 사용하고 있지 않는데 이는 Well-known Port 번호를 사용하는 애플리케이션을 대상으로 트래픽 분류를 하기 때문에 Port 번호는 Feature에서 제외한 것으로 보인다. 기존 연구들은 Cross Validation을 적용하여 분류하였으며, 분류 결과는 92~93%의 정확도를 가지고 있다.

Feature	Erman et al. [2]	Zander et al. [3]	Nguyen et al. [4]	Park et al. [5]	Andrew et al. [6]	Erman et al. [7]	Williams et al. [8]
Total number of packet	O			O		O	
Packet Size	O		O	O		O	O
Payload Size	O				O		
Number of transferred bytes	O						
Inter packet arrival time	O	O	O	O	O	O	O
IP address		O	O		O		
Port Number		O	O		O		O
Flow size		O					O
Flow duration		O		O	O	O	O
Protocol			O				
Inter packet length variation			O				
Initial advertised-window bytes				O			
Number of packet with 'PUSH' option				O			
Advertised-windows bytes				O			
Effective Bandwidth based upon entropy					O		
Fourier Transform of packet					O		

표 2 기존 연구에서 사용된 Feature

Erman et al.[2]은 K-Means, DBSCAN, AutoClass와 같은 Unsupervised 알고리즘을 이용하여 애플리케이션 트래픽 분류를 수행하고 있다. 이 논문[2]에서는 각 ML 알고리즘을 통한 애플리케이션 트래픽 분류 결과를 비교 분석하여 보여준다. 오클랜드와 캘거리 대학의 인터넷 링크에서 수집한 데이터를 이용하여 단일한 Port 번호를 사용하는 애플리케이션을 대상으로 트래픽을 분류한 결과를 보여주고 있다. 이는 곧 트래픽 분류의 전통적인 방법인 Port 별 분류를 하는 것과 다르지 않은 결과를 보여준다고 할 수 있다. 본 논문에서는 POSTECH의 인터넷 링크에서 수집한 데이터에 대해 Port를 두 개 이

상 동적으로 할당 받아 사용하는 애플리케이션도 하나의 같은 애플리케이션으로 분류한다.

Zander et al.[3]은 AutoClass 알고리즘을 이용하여서 여러 네트워크 망에서 수집한 트래픽을 애플리케이션 별로 나누어 주는 결과를 보여준다. 이 논문[3]은 각 Feature가 애플리케이션 트래픽 분류에 끼치는 영향을 측정하여서 보여주고 있다. 하지만 이 분류 결과도 단일한 Port 번호를 사용하는 애플리케이션의 트래픽을 분류하고 있다.

Nguyen et al.[4]는 Short Sub-Flows의 Combination을 Training 하는 기법을 제시하고 있다. 25개의 Packet을 하나의 Short Sub-Flows로 정의하고 다수개의 Sub-Flows를 Combination 하여서 Training을 시키는 방법이 좋은 분류 결과를 얻을 수 있음을 보여준다. 이 논문[4]에서 사용한 알고리즘은 Naïve Bayes 알고리즘이다.

Park et al.[5]은 Naïve Bayes, Naïve Bayes Kernel Estimator와 REPTree 알고리즘을 이용하여 애플리케이션을 QoS Provision에 따라 구별하고 QoS 수준 별로 트래픽 분류가 이루어지는 Scheme을 소개하고 있다. 또한, 하나의 QoS가 아닌 애플리케이션 별로 트래픽 분류를 한다. Park et al.은 또한 다른 논문[9]에서 Feature Reduction 알고리즘인 Genetic 알고리즘과 J48, Naïve Bayes Kernel Estimator, REPTree인 ML 알고리즘을 적용한 애플리케이션 트래픽 분류의 결과를 비교 분석하여 Feature Reduction 알고리즘이 가지는 효과를 보여준다.

Moore et al.[6]는 Naïve Bayes, Naïve Bayes Kernel Estimator를 이용해서 애플리케이션 트래픽 분류를 수행한다. 분류 결과의 비교 분석을 통하여 Bayesian Analysis Technique가 애플리케이션 트래픽 분류를 하기에 가장 성능이 좋은 ML 알고리즘임을 보였다. 본 논문에서는 [6]에서 다루지 않은 Neural Network 알고리즘을 추가하여 4가지의 ML 알고리즘을 이용하여 애플리케이션 트래픽 분류를 수행한 결과를 비교 분석한다.

Erman et al.[7]은 EM 알고리즘을 이용하여 애플리케이션 트래픽 분류를 하였다. 그리고 Supervised ML 알고리즘인 Naïve Bayes 알고리즘을 이용한 애플리케이션 트래픽 분류 결과와의 비교를 통하여 Unsupervised ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 더 우위임을 보여주었다.

Williams et al.[8]은 Naïve Bayes, C4.5, Bayesian Network, Naïve Bayes Tree등 5가지 ML 알고리즘과 여러 가지 Feature Reduction 알고리즘을 적용하여 애플리케이션 트래픽을 분류한다. 이 논문에서는 5가지 알고리즘 중에 좋은 성능을 보이는 알고리즘으로 C4.5와 NBTree를 제시하고 있으며, 2가지 Feature Reduction 알고리즘을 적용한 것과 적용하지 않은 것을 비교 분석하여 꼭 필요한 Feature들을 선택하여서 수행시간을 줄여주는 측면에서의 Feature Reduction 알고리즘의 필요성을 보여준다. 본 논문에서 우리는 [8]에서 다루지 않은 Neural Network인 Multi-layer Perceptron 알고리즘을 포함한 4가지 ML 알고리즘을 적용한 애플리케이션 트래픽 분류 결과를 비교 분석하여 가장 좋은 ML 알고리즘을 제시한다.

Park et al.[9]는 NBKE, J48, REPTree 알고리즘을 적용하여 애플리케이션 트래픽 분류를 수행한 결과를 보여준다, 그리고 Feature Reduction 알고리즘으로 FCBF 알고리즘과 GA 알고리즘을 적용했을 때와 Feature Reduction 알고리즘을 적용하지 않았을 때의 애플리케이션 트래픽 분류 결과를 비교함으로써, Feature Reduction 알고리즘의 필요성을 강조한다.

마지막으로, 기존 연구 논문에서 애플리케이션 트래픽 분류를 제시하는 기준은 크게 두 가지로 나누어 볼 수 있다. 표 3는 (1) Well-known Port 번호를 사용하거나 고정된 Port 번호를 사용하는 애플리케이션 별로 트래픽 분류 결과를 제시하는 것과 (2) 동일한 애플리케이션들을 성격에 맞게 Grouping하여 Category한 것을 기준으로 분류한

결과를 보여준다. 여기서 고정된 Port 번호를 사용한다는 의미는 애플리케이션이 자신의 정체를 숨기기 위해서 80번과 같은 Well-known Port 번호를 사용하는 것을 말하는 것이 아니라 IANA에서 정의한 Well-known Port 번호를 사용하거나, 다른 애플리케이션에서 사용하지 않는 Port 번호 중 하나를 골라서 사용하는 것을 말한다.

Author	Classification	Description
Andrew et al. [6]	BULK, DATABASE, INTERACTIVE, MAIL, SERVICE, WWW, P2P, ATTACK, GAMES, MULTIMEDIA	단일한 Port 번호를 사용하는 애플리케이션 Category 별로 분류
Park et al. [5]	INTERACTIVE, BULK, REAL TIME, EMAIL, TRANSACTION	
Erman et al. [2]	DNS, FTP, HTTP, IRC, LIMEWIRE, NNTP, POP3, SOCKS.	단일한 Port 번호를 사용하는 특정 애플리케이션 별로 분류
Williams et al. [8]	FTP, TELNET, SMTP, DNS, HTTP, Half-Life	

표 3. 기존 연구에서 제시한 애플리케이션

먼저, Well-known Port 번호를 사용하거나 또는 고정된 Port 번호를 사용하는 애플리케이션 트래픽 분류 결과를 제시하는 기존 연구[2, 8]를 보면, 분류 결과를 보여주는 대표적인 애플리케이션으로는 FTP-Data(Port 20), FTP-Control(Port 21), Telnet(Port 23), SMTP(Port 25), DNS(Port 53), HTTP(Port 80), Half-Life(Port 27015)를 들 수 있다. 이 외에 Well-known Port 번호를 가지는 트래픽을 발생하는 여러 Vendor들의 DBMS나 MAIL 서버도 하나의 애플리케이션으로 분류되고 있으며, 네트워크 장비의 제어를 위한 트래픽도 Protocol(Port) 별로 분류 결과를 제시한다. 그리고 P2P 애플리케이션 중 대표적인 Port 번호로 6346번을 사용하는 Gnutella의 애플리케이션인 LIMEWIRE도 분류하고 있으며, 대표적인 Port 번호로 1214번을 사용하는 KaZaA와 같은 것도 분류한다.

다음으로 Well-known Port 번호를 사용하거나 또는 고정된 Port

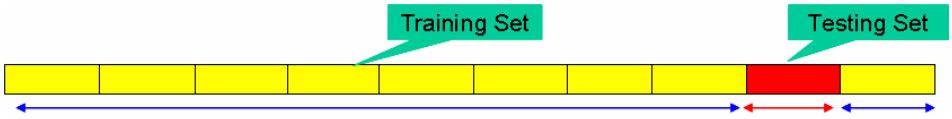
번호를 사용하는 애플리케이션들을 성격에 맞게 Grouping하여 Category하여 분류 결과를 제시하기도 하였다[5, 6]. 예를 들어 [6]은 postgres, sqlnet, oracle, ingres는 ‘DATABASE’, X11, dns, ident, ldap, ntp는 ‘SERVICE’, 그리고 ssh, klogin, rlogin, telnet은 ‘INTERACTIVE’로 Category 하여서 분류 결과를 제시하였다.

즉, 기존 연구의 애플리케이션 트래픽 분류 결과는 Dynamic한 Port 번호를 사용하거나 또는 정체를 숨기기 위해 Well-known Port 번호를 사용하는 애플리케이션이 많이 존재하는 현재의 네트워크의 환경을 반영하지 못한 것이다. 본 논문에서는 현재의 네트워크의 환경에 적합한 애플리케이션 트래픽 분류 결과를 제시한다.

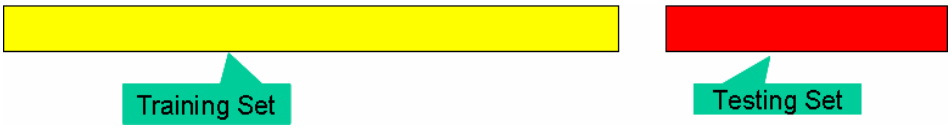
2.3 평가 방법

ML 알고리즘을 이용한 애플리케이션 트래픽 분류에서 고려해야 할 평가 방법은 Cross Validation과 Split Validation의 두 가지가 있다. Cross Validation은 동일한 시간대에 수집된 하나의 데이터 Set 안에서 Training Set과 Testing Set을 구성한다. 일반적으로 하나의 데이터 Set을 10등분 하여 그 중 하나를 Testing Set을 위한 데이터로 사용하고 나머지 9개를 Training Set을 위한 데이터로 사용한다. Testing Set이 10번 바뀌고 동시에 Training Set도 10번 바뀌면서 10번의 실험이 이루어 진다. 반면에 Split Validation은 각각 다른 시간대에서 수집한 두 개의 데이터를 이용하여서 Training Set과 Testing Set을 구성한다. 즉, Training Set과 Testing Set이 독립적인 데이터로 이루어진다.

그림 1의 (a)와 (b)는 Cross Validation과 Split Validation이 Training Set과 Testing Set을 구성하는 차이를 잘 보여준다.



(a) Cross Validation



(b) Split Validation

그림 1. Cross Validation과 Split Validation의 차이

ML 알고리즘을 통한 애플리케이션 트래픽 분류가 잘 되었는가 평가하기 위해서 Precision과 Recall 그리고 Overall Accuracy 세 가지 평가 기준이 있다. Overall Accuracy는 전체 데이터를 하나로 놓고 제대로 분류가 된 양이 얼마나 되는지에 대해서 알아보기 위한 평가 기준이다. 반면에 Precision과 Recall은 각 애플리케이션을 기준으로 분류가 얼마나 정확하게 되었는지에 대해서 알아보기 위한 평가 기준이다.

Recall과 Precision을 예를 들어 설명하면 다음과 같다. 실제 A라는 그룹에 속하는 30개 데이터를 ML 알고리즘을 이용해서 분류 한 결과 그 중에서 A 그룹에 속했다고 분류한 결과 값이 40개이고 이중 실제 A 그룹에 속하는 데이터가 20개였다면, 전체 A 그룹의 30개 중 20개만 A 그룹에 속했다고 분석한 비율인 20/30이 Recall의 값이다. 그리고 분류한 결과 A라는 그룹에 속한다고 나온 결과 40개 중에는 A 뿐 아니라 B나 C 그룹 내의 값도 포함된 것이다. 제대로 분류한 A 값은 20개 이므로 A 그룹으로 분류된 데이터 중 실제로 A에 속하는 데이터의 비율은 20/40, 즉 제대로 분류한 비율 값이 Precision이다.

Recall은 'ML 알고리즘을 적용하여 A 그룹으로 분류된 원소 중 실제 A 그룹에 속한 개수 / 분류 전 실제 데이터 Set에서 A 그룹의

원소 개수'이고 Precision은 'A 그룹으로 분류된 원소 중 실제 A 그룹에 속하는 원소 개수 / A 그룹으로 분류된 전체 원소 개수'이다. Precision과 Recall 그리고 Overall Accuracy는 True Positive(TP)와 False Positive(FP), 그리고 False Negative(FN) 등으로 아래와 같은 식으로 나타낼 수 있다[7].

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{(식 1)}$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{(식 2)}$$

$$\text{Overall Accuracy} = \frac{\sum \text{TP of each Application}}{\text{Total element}} \quad \text{(식 3)}$$

애플리케이션 트래픽 분류를 위해서, 본 논문에서 적용하는 4가지 알고리즘과 4가지 Feature Set 중 가장 좋은 ML 알고리즘과 Feature Set을 찾기 위해 (식 3)을 이용하여 Overall Accuracy 값이 가장 높은 것을 선택한다. 또한 가장 높은 정확도를 가지는 ML 알고리즘과 Feature Set의 Match에 대해서는 각 애플리케이션의 Precision과 Recall 도 살펴보아 분류의 정확도를 알아보고자 한다.

2.4 문제점 및 해결 방안

기존 연구를 살펴보면 세 가지 문제점을 생각해 볼 수 있다. 그리고 이 문제점을 해결할 수 있는 방안을 제안함으로 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 실제 Internet Link상에서 가능하도록 한다.

2.4.1 Port 번호 기반의 애플리케이션 분류 결과

ML 알고리즘을 적용한 애플리케이션 트래픽 분류의 기존 연구는 애플리케이션들이 동적으로 Port 번호를 할당하여 수행되는 것에 대해서는 그 특성을 전혀 반영하고 있지 않다. 이렇듯 변하는 애플리케이션의 성격을 전혀 고려하지 않은 채, Well-known Port 번호를 포함한 고정된 Port 번호를 사용하는 애플리케이션을 선택하여 그것들 별로 분류한 결과를 보여주고 있다. 또한, 이러한 애플리케이션 트래픽 분류 결과를 이용해서 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 Dynamic하게 변화하고 있는 애플리케이션의 성격에 대응하는 트래픽 분류 방법이라고 말하기 힘들다. Port 기반이 아닌 실제로 애플리케이션이 발생하는 트래픽을 수집하여 Training을 시킨 후에 그 훈련 결과를 Testing에 이용함으로써 기본 ML 알고리즘을 적용한 애플리케이션 트래픽 분류 연구들이 Port 번호 기반의 분류를 벗어나지 못하는 한계를 극복할 수 있다. 또한, 본 논문에서는 Well-known Port 번호인 80번을 기본 Port 번호로 사용하는 P2P나 Web Disk 애플리케이션을 포함하며 Dynamic하게 변하는 다수의 애플리케이션을 포함한 인터넷 트래픽 분류 결과를 제시함으로써 최근 애플리케이션의 변화를 반영하는 ML 알고리즘을 적용한 인터넷 애플리케이션 트래픽 분류 결과를 제시한다.

기존의 논문들이 애플리케이션 분류 기준으로 Port 번호를 적용한 까닭은 분류 대상의 트래픽 데이터가 어떤 애플리케이션에서 발생한 것인지에 대한 배경 지식을 알 수 없기 때문이다. 기존 논문들은 배경 지식 없이 애플리케이션 트래픽 분류를 하기 위해서 트래픽 Header에서 쉽게 얻을 수 있는 Port 번호를 기반으로 Well-known Port 번호만 사용하는 애플리케이션들을 대상으로 분류 결과를 제시하였다.

본 논문은 POSTECH의 Internet Link의 트래픽 데이터를 대상으

로 애플리케이션 트래픽 분류를 수행하기 위해서 동일한 트래픽 데이터를 분류하는 NGMON의 애플리케이션 트래픽 분류 결과를 배경 지식으로 활용하여 P2P나 Web Disk 등 다양한 애플리케이션을 선정하였다. 그리고 각 애플리케이션이 생성하는 트래픽을 실제로 수집하여 Training Set과 Testing Set을 만들고 이를 기반으로 ML 알고리즘을 적용한 애플리케이션 트래픽 분류를 수행하였다.

2.4.2 한정된 데이터 안에서의 분류

기존 연구[2, 3, 4, 7, 8]를 살펴보면 Training과 Testing을 위한 기법으로 Cross Validation 기법을 선택하고 있다. Cross Validation 기법을 선택하는 것은 Closed Set안에서 Training을 시키고 Testing을 수행함으로써 의도하지 않은 문제점을 야기한다. Cross Validation은 Training Set과 Testing Set이 동시간에 얻어진 트래픽 데이터로 구성된 것이기 때문에 Training Set과 Testing Set이 가지는 Pattern이 같다. 그래서 애플리케이션 트래픽 분류 결과를 측정하는 Overall Accuracy가 매우 높다. 예를 들어, 애플리케이션이 사용하는 Port 번호를 보면, 대부분의 애플리케이션은 더 이상 고정적인 포트 번호를 사용하지 않는다. 특히 애플리케이션을 사용하는 Client에서 사용하는 포트 번호는 유동성의 정도가 더 심하다. Client 애플리케이션의 포트 번호는 특정한 Seed 값에서 1씩 증가하면서 할당되는 특징을 가지고 있다. 특정 시간에 특정한 애플리케이션 'A'만을 사용한 데스크톱에서 얻은 데이터를 살펴보니 Client에서 할당한 포트 번호가 1,000번부터 3,000번까지 1씩 증가하는 것을 볼 수 있다. 이 데이터를 Cross Validation 기법에 대입해 보면, 그 데이터 Set 안에서 Training을 위한 데이터와 Testing을 위한 데이터가 형성되므로, 이렇게 형성된 데이터들을 가지고 애플리케이션 트래픽 분류를 하게 되면 1~3000번에 있는 포트 번호를 가지는 데이터는 'A'라는 애플리케이션에 의해서 발생된 것이라고 분류되기 쉽고

그 외의 포트 번호를 가지는 데이터는 ‘A’라는 애플리케이션에 의해서 발생된 것이라고 분류되기 어렵다.

반면에 Split Validation을 이용하여 애플리케이션 트래픽 분류를 할 때, 앞에서 말한 데이터를 Training을 위한 데이터로 사용하고 ‘A’ 애플리케이션을 다른 시간대에 수행하여 수집하여 Client Port 번호가 6,000번에서 9,000번까지로 형성된 데이터를 가지고 Testing을 하게 되면 이것은 ‘A’라고 분류되기 어렵다. 즉, Cross Validation을 이용하여 분류를 한 결과만큼 분류의 정확도가 높게 나오지 않는다. Split Validation은 Training Set과 Testing Set이 다른 시간에 얻어진 트래픽 데이터를 구성된 것이기 때문에 Training Set과 Testing Set이 가지는 Pattern이 항상 같지 않다. 즉, Cross Validation을 이용한 애플리케이션 트래픽 분류를 기반으로 하는 기존의 연구 결과가 Internet Link의 실제 네트워크 트래픽 모니터링 환경의 네트워크 애플리케이션을 분류하는데 적용하기에는 부적절한 면이 있다. 즉, Closed 데이터 Set에서의 Training과 Testing을 수행하여 높은 정확도를 보이는 ML 알고리즘과 Feature Set을 Internet Link의 애플리케이션 트래픽 분류에 적용하기에는 Testing 환경이 다르기에 적합하지 않다는 것을 알 수 있다. 그림 2는 표 6에서 볼 수 있는 Feature들 전체로 구성한 Feature Set(1:all)과 표 6에서 볼 수 있는 Feature들 중 IP Address를 제외한 Feature들로 구성한 Feature Set(2:without IP)을 이용하고 ML 알고리즘으로 J48 알고리즘을 적용한 애플리케이션 트래픽 분류 결과로 Overall Accuracy를 보여준다. 애플리케이션의 특성을 반영하기에 충분치 않은 데이터 Set을 이용하여 Training을 했을 때에 Split Validation으로 애플리케이션 트래픽 분류한 결과와 Cross Validation으로 애플리케이션 트래픽 분류한 결과는 두가지 Feature Set을 적용했을 때 분류의 정확도는 Cross Validation을 적용하면 약 95%, 96%이지만 Split Validation을 적용하면 65%, 66%가 나옴을 알 수 있다. 즉, 충분치 않은 데이터를 이용해도 Cross Validation을 적용하면 좋은 정확도를 얻을 수 있으며, 이는 적합

한 결과가 아님을 알 수 있다.

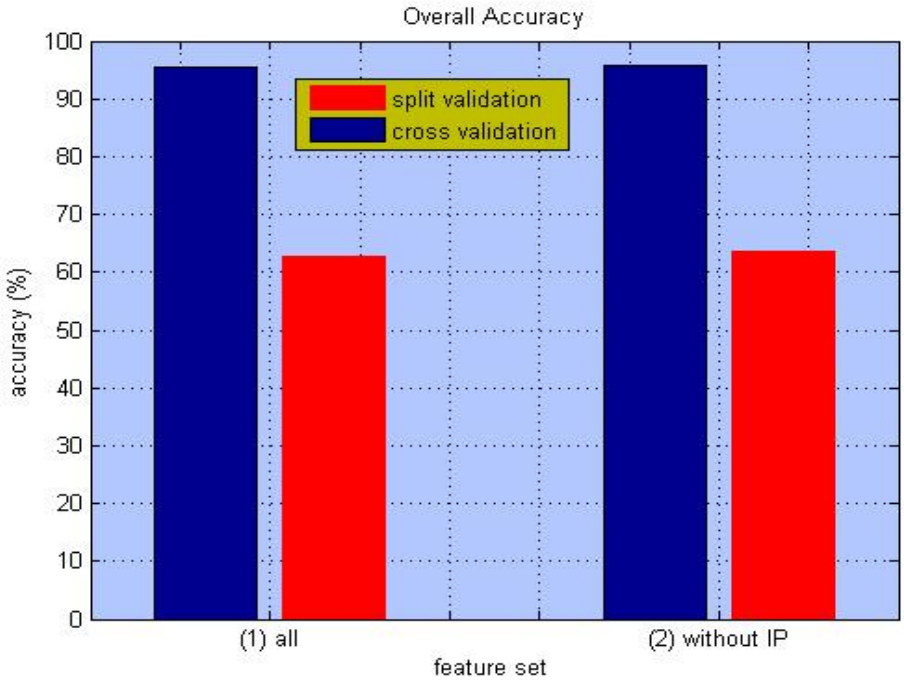


그림 2. Cross Validation과 Split Validation의 Overall Accuracy

2.4.3 Flow 기반의 분류

대부분의 논문[4, 5, 6, 8, 9]은 Flow 기반으로 애플리케이션 트래픽 분류의 정확도를 측정한 반면에 [8]은 Byte를 기반으로 애플리케이션 트래픽 분류의 정확도를 측정하고 있다. 대부분의 기존 연구에서는 Feature Set을 구성하는 정보가 Flow 기반에서 얻을 수 있는 것들로 구성되어 있고, 이러한 Feature Set을 통해서 Training과 Testing을 수행하고 있다. 즉, Training과 Testing을 위한 데이터 Set이 Flow 기반의 정보에서 추출되고, 그 애플리케이션 트래픽 분류 결과의 정확도 또한 Flow 기반 데이터 값에 의해서 평가된다.

그러나 Flow 기반의 데이터를 기반으로 정확도를 측정하는 경우에, 여러 가지 문제가 발생할 수 있다. 첫째, 각 Flow가 전체 트래픽 양에서 차지하는 비중은 Flow마다 다르다. 즉, Flow가 가지고 있는 Packet의 수가 다르고 전체 Byte 양도 다를 수 있다. 예를 들면, 어떤 Flow는 10,000 Byte로 이루어져 있고, 다른 Flow는 1,000 Byte로 이루어져 있으면, 각각이 한 개의 Flow지만 Byte 기준으로 보면 이 두 개의 Flow가 제대로 분류 되었는지 정확도를 측정함에 있어서 그 비중이 분명히 다르다는 것이다. 이러한 문제는 대용량 Byte를 많이 발생시키는 P2P 애플리케이션 또는 Web disk, FTP 애플리케이션에서 많이 찾아볼 수 있다. 이러한 문제를 Class Imbalance Problem이라고 한다[11]. 여기서 Class는 하나의 애플리케이션으로 볼 수 있는데, 대용량의 Byte를 발생시키는 애플리케이션과 MSN Messenger, Google Talk와 같은 작은 양의 Byte를 발생시키는 Chatting 애플리케이션은 모든 애플리케이션에서 발생하는 Byte에서 차지하는 비중 측면에서 볼 때, 분명 그 양에서 상대적인 차이를 보이게 된다.

또한 Flow만을 기반으로 애플리케이션 트래픽 분류를 수행하는 것은 네트워크 모니터링에서 중요한 부분을 차지하는 Traffic Shaping이나 Usage Billing Policy에 적용하기 위한 실제적인 트래픽 데이터 양에 관련한 정확한 데이터를 제공하지 못하기 때문에[11] 이러한 문제를 해결하기 위해서라도 애플리케이션 트래픽 분류 결과의 정확도를 Flow 기반과 함께 Byte 기반으로 고려하는 것도 필요하다.

그림 3은 6가지의 ML 알고리즘(J48, REPTree, Bayesian-Network, Naïve-bayesian, Multi-layer Perceptron, RBFNetwork)을 적용하고 표 6에 있는 모든 Feature들로 구성한 Feature Set을 이용한 애플리케이션 트래픽 분류의 결과로 Overall Accuracy을 Byte를 기반으로 측정한 것과 Flow 기반으로 측정한 것을 제시하고 있다. 이 결과를 살펴보면, J48은 Byte를 기반으로 분류를 수행하면 Overall Accuracy가 85% 정도 이지

만 Flow 기반일 경우에는 약 63% 정도의 정확도를 보인다. 대부분의 알고리즘의 차이가 약 6%에서 많으면 약 20%까지 보이면서 Byte를 기반으로 했을 때에 정확도 결과가 더 높음을 알 수 있다. 그러나 RBFNetwork 알고리즘은 Byte를 기반으로 했을 때에는 약 78%이지만 Flow를 기반으로 하면 약 81%의 정확도가 나온다. 또한 BayesNet 알고리즘도 Byte를 기반으로 했을 때에는 정확도 값이 약 77%였지만 Flow를 기반으로 하면 약 82%의 정확도가 나온다 이와 같이 근소한 차이를 보이면서 Flow 기반으로 분류했을 때에 Overall Accuracy값이 더 높은 결과가 나오는 ML 알고리즘도 있다.

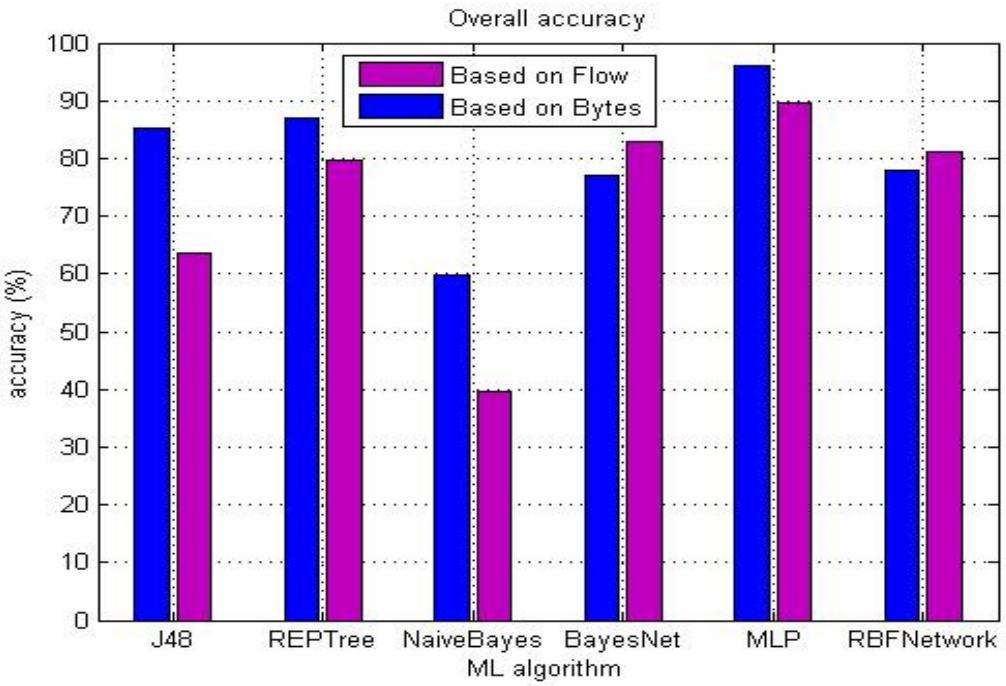


그림 3. Byte와 Flow 기반의 Overall Accuracy

3 ML 알고리즘 적용 방안

이 장에서는 2.4장에서 제안한 해결 방안을 기반으로 ML 알고리즘을 적용한 애플리케이션 트래픽 분류 방안을 제시한다.

3.1 분류 애플리케이션 선정

본 논문은 2.4.1장에서 언급 하였듯이, 80번을 기본 Port 번호로 사용하는 P2P 애플리케이션이나 Web Disk 애플리케이션을 포함하고 Port 번호가 Dynamic하게 변하는 애플리케이션을 포함하였다. 표 4는 본 논문에서 애플리케이션 트래픽 분류하려는 애플리케이션을 보여준다. 이탤릭체로 쓰여진 애플리케이션은 Dynamic한 Port 번호를 사용하는 애플리케이션이고, 밑줄이 그어진 애플리케이션은 기본 Port 번호로 80번을 사용하는 애플리케이션들이다. 나머지 애플리케이션은 정해진 고정 Port 번호를 사용하는 애플리케이션이다.

Category	Application
Web Disk	*CLUBBOX*, PARANDISK, TOTODISK, ENDISK
Web Traffic	HTTP, HANGAME, SAYCLUB
FTP	ALFTP
P2P	*FILEGURI*, SORIBADA, GAMPLE, *BITTORRENT*
Game Traffic	STARCRAFT
Remote Controller	MSTC

표 4. POSTECH의 애플리케이션 Trace

Web Traffic은 HTTP, HTTPS 프로토콜을 포함한 웹 브라우저를 이용한 대부분의 애플리케이션을 의미한다. 그리고 Web disk는 자료를 토탈 사이트에서 제공하는 저장 공간에 저장하고 여러 사용자들이 그

자료를 공유하는 애플리케이션이다. 본 논문은 CLUBBOX, PARANDISK, TOTODISK, ENDISK의 Web Disk 애플리케이션을 분류한다. 또한 FTP 프로토콜을 사용하는 애플리케이션인 ALFTP를 분류한다. P2P 애플리케이션 중 FILEGURI, BITTORRENT, GAMPLE, SORIBADA를 분류한다. P2P 애플리케이션 중 BITTORRENT는 애플리케이션 내에서 파일 검색을 하지 않고, 웹 기반의 검색 엔진을 이용하여 파일 검색을 한다. 본 논문에서는 파일 검색을 하는 웹 트래픽도 BITTORRENT의 트래픽으로 간주한다. GAMPLE과 FILEGURI는 유료 P2P 애플리케이션이다. 게임 애플리케이션으로 STARCRAFT의 Battle-net도 분류 대상이 된다. MS Windows에서 제공하는 Remote Computer Controller인 MSTC도 하나의 애플리케이션으로 분류한다.

표 5는 본 논문에서 사용한 데이터의 용량과 데이터를 Flow로 변환했을 때 Flow의 개수를 나타낸다. STARCRAFT의 경우 게임 트래픽의 특성상 발생한 데이터가 적어 데이터 수집이 용이치 않아, Training과 Testing에 사용한 Flow 데이터 수가 적다.

	BITTORRENT		FTP		CLUBBOX		DESKTOP		FILEGURI		GAMPLE		ENDISK	
	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)
Training	11000	708	518	1011	231	165	86	28	571	506	10943	989.9	246	447
Testing	4000	256	231	342	165	102	34	12.4	408	407	546.9	4032	550	384

	HANGAME		HTTP		PARANDISK		SAYCLUB		SORIBADA		STARCRAFT		TOTODISK	
	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)	Flow(개수)	Byte(MB)
Training	998	23.5	4263	26.8	223	278.45	2870	10.07	8803	339	15	0.244	526	367
Testing	47	7	2045	15.3	148	145	85	3	5960	273	6	0.126	247	192

표 5. 각 애플리케이션 데이터 size 및 Flow 개수

3.2 데이터 수집 방법

데이터 수집은 특정 호스트의 특정한 애플리케이션에서 발생한 트래픽 데이터를 Ethereal[14]를 이용하여 호스트에서 수집하고 특정 호스트로부터 발생한 트래픽 데이터를 인터넷 링크에서 DAG 카드를 가지는 트래픽 Capture 시스템을 이용하여 수집하였다. 호스트에서 수집한 데이터는 Training Set을 구성하기 위한 데이터로 사용하고, 인터넷 링크에서 수집한 데이터는 Testing을 위한 데이터로 사용되었다. 즉, 호스트로부터 얻은 데이터를 기반으로 한 Training Set을 이용하여 Training을 한 모델을 이용하여 인터넷 링크에서 수집한 애플리케이션 트래픽을 기반으로 한 Testing Set을 분류한다. 그림 4는 본 논문에서 트래픽을 수집한 위치를 보여준다.

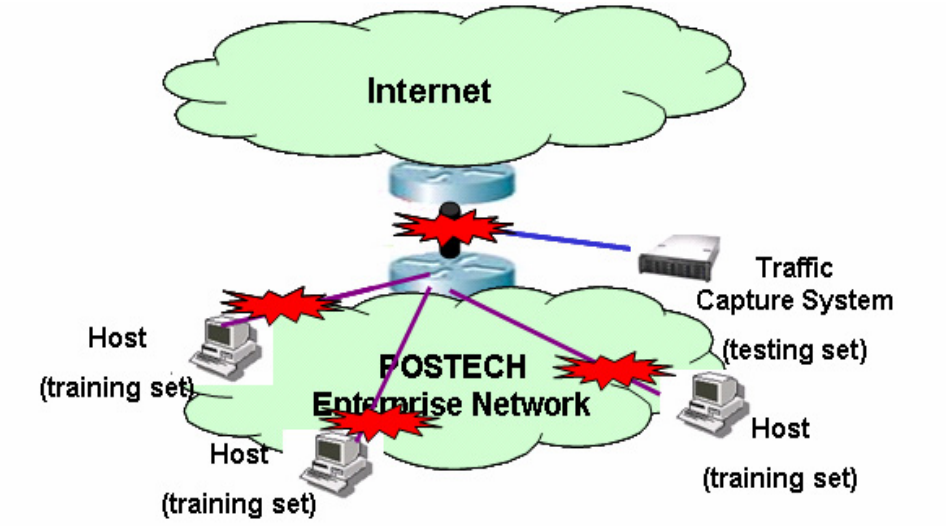


그림 4. 트래픽 Capture 위치

3.3 Split Validation 기법 적용

본 논문은 2.4.2장에서 언급하였듯, Internet Link상의 Real-time 트래픽 데이터의 특성상 Split Validation 기법을 적용한 ML 알고리즘을

적용한 애플리케이션 트래픽 분류를 수행한다, Split Validation으로 애플리케이션 트래픽 분류를 하기 위해서 약 10시간 동안 Host에서 수집한 각 애플리케이션의 데이터를 기반으로 Training Set을 만들었다, 또한, 본 논문은 한 애플리케이션이 제대로 Training이 되려면 적어도 25개 이상의 Flow가 필요하다는 [4]의 연구 결과를 참고하여서 트래픽을 수집하였다.

3.4 ML 알고리즘과 Feature Set 선정

본 논문에서 애플리케이션 트래픽 분류에 적용한 ML 알고리즘은 기존 논문에서 많이 적용한 J48, REPTree, BayesNet과 기존 논문에서 드물게 사용했지만 애플리케이션 트래픽 분류 이외의 다른 영역에서 충분히 좋은 성능을 보이는 Multi-layer Perceptron의 4가지 ML 알고리즘이다.

또한, ML 알고리즘을 이용하여 애플리케이션 트래픽 분류의 정확도를 높이기 위해서는 데이터의 특성을 나타낼 수 있는 Feature를 잘 선정해야 한다. ML 알고리즘을 이용한 애플리케이션 트래픽 분류를 한 기존 연구는 Feature를 잘 선정하기 위해 Session 기반 또는 Flow 기반으로 Packet 데이터를 Aggregation한다. Session을 기반으로 Aggregation해서 Feature를 선정하는 것은 UDP 트래픽 데이터를 다루기가 어렵다는 단점이 있다. 본 논문에서는 UDP 트래픽 데이터를 발생하는 게임 트래픽을 다루기 위하여 Source IP Address, Source Port 번호, Destination IP Address, Destination Port 번호, Protocol로 정의한 Flow 기반으로 트래픽 데이터를 Aggregation 한다. 표 6은 본 논문에서 사용하는 Feature를 보여준다.

IP address (source, destination)
Port number (source, destination)
Byte counts
Connection duration
Packet size statistics (minimum, maximum, mean, standard deviation)
Inter-packet arrival time statistics (minimum, maximum, mean, standard deviation)

표 6. 선정한 Feature

Packet header에서 라우팅 정보로 사용되는 IP Address와 Port 번호, 그리고 하나의 Flow안에 속한 Packet들의 Byte 총합을 나타내는 Byte Count, 그리고 Flow가 유지된 시간을 의미하는 Connection Duration, 그리고 하나의 Flow에 속하는 Packet의 Size Distribution과 Inter-packet Arrival Time의 Distribution인 Feature들을 이용하여 Feature Set을 구성한다.

기존의 다른 연구에서는 IP Address와 Port 번호를 사용하거나 또는 사용하지 않는다. 본 논문에서는 IP Address와 Port 번호를 사용할 때와 그렇지 않을 때를 반영한 4가지의 Feature Set을 이용한 애플리케이션 트래픽 분류의 정확도를 비교 분석해 봄으로써 4가지 중 가장 좋은 Feature Set을 제시하고자 한다. 4가지의 Feature Set은 다음과 같이 정의한다.

(1) all: 모든 Feature이 선택된 경우
(2) without IP: 모든 Feature에서 Source와 Destination의 IP Address를 제외한 경우
(3) without Port: 모든 Feature에서 Source와 Destination의 Port 번호를 제외한 경우
(4) without IP&Port: 모든 Feature에서 Source와 Destination의 IP Address와 Port 번호를 제외한 경우

3.5 ML 알고리즘을 적용한 분류 결과

이 장에서는 3.4장에서 언급한 4가지 Feature Set과 애플리케이션 트래픽 분류한 애플리케이션 트래픽 분류를 수행하고, 그 결과를 비교 분석한다. 그리고 분류의 Overall Accuracy를 가장 높일 수 있는 ML 알고리즘과 Feature Set을 제시한다. 3.2장에서 언급한 것처럼 애플리케이션 트래픽 분류 하기 위해 호스트에서 수집한 데이터를 기반으로 한 Training Set을 Modeling한 것을 기반으로 인터넷 링크에서 수집한 데이터를 기반으로 한 Testing Set을 분류하였다. 이러한 분류 결과를 Flow 기반과 Byte 기반으로 측정한 Overall Accuracy 값을 보여준다.

3.5.1 Flow 기반의 분류 결과 분석

그림 5는 4가지 ML 알고리즘과 4가지 Feature Set을 적용한 애플리케이션 트래픽 분류의 결과를 Flow 기반으로 측정한 Overall Accuracy 값을 보여준다. J48과 BayesNet 알고리즘을 4가지 모든 Feature Set 정보를 이용하여 분류에 적용한 결과 90%에 가까운 Overall Accuracy를 보여준다.

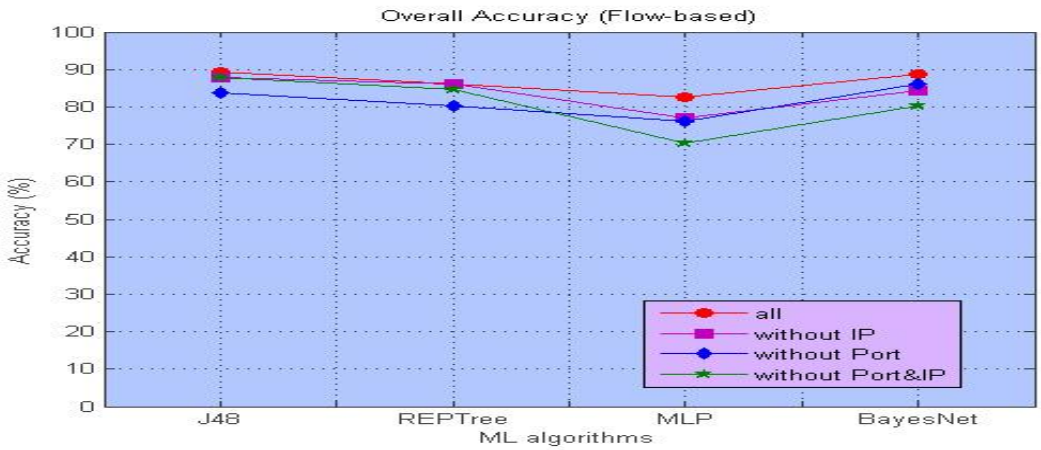


그림 5. Flow 기반 Overall Accuracy

표 7은 가장 좋은 애플리케이션 트래픽 분류 결과를 보이는 BayesNet 알고리즘을 적용하고 3.4장에서 정의한 모든 Feature들로 구성한 (1)번 Feature Set을 적용하여 Flow 기반으로 각 애플리케이션 트래픽 분류를 한 결과를 2.3장에서 정의한 Precision과 Recall로 나타낸다.

Application	Precision	Recall
BITTORRENT	0.978	0.757
ALFTP	0.177	0.061
CLUBBOX	0.615	0.508
DESKTOP	1	1
FILEGURI	0.713	0.559
GAMPLE	0.962	0.983
HANGAME	0.169	0.511
HTTP	0.917	0.956
PARANDISK	0.185	0.432
SAYCLUB	0.126	0.376
SORIBADA	0.891	0.983
STARCRAFT	1	0.231
TOTODISK	0.784	0.514

표 7. Flow 기반의 애플리케이션 별 Precision & Recall

표 7을 살펴보면 ALFTP와 PARANDISK, HANGAME, SAYCLUB의 분류 결과 Precision과 Recall 값이 좋지 않음을 알 수 있다. STARCRAFT는 Precision은 100%로 매우 높지만 Recall은 약 23%로 낮다. 즉, STARCRAFT라고 분류된 데이터는 100% 제대로 분류가 되었지만 실제로 STARCRAFT 트래픽임에도 불구하고 분류가 되지 못한 데이터가 많다는 것이다. 이는 Training Set으로 사용하기 위해 모은 데이터의 양이 Testing Set의 데이터가 가지는 모든 특성을 반영하지 못하기 때문이다.

ALFTP의 Precision은 약 17%이고, Recall은 약 6%이며 PARANDISK의 Precision과 Recall은 약 18%와 43%로 분류 결과가 좋

지 않다. 이 까닭은 ALFTP와 PARANDISK의 데이터가 가지는 특성이 매우 비슷하여 ALFTP의 많은 데이터가 PARANDISK의 데이터로 분류가 되었으며 PARANDISK의 많은 데이터가 ALFTP의 데이터로 분류되었기 때문이다. 분류 결과의 Confusion Matrix를 살펴보면 ALFTP에 속하는 약 60%의 데이터가 PARANDISK로 분류가 되었고 PARANDISK에 속하는 약 40%의 데이터가 ALFTP로 분류되었다.

HANGAME의 분류 결과는 Precision은 약 17%, Recall은 51%이고 SAYCLUB의 분류 결과는 Precision은 약 12%, Recall은 약 36%이다. HANGAME과 SAYCLUB은 웹 브라우저 안에서 동작하는 애플리케이션이기 때문에 많은 데이터가 HTTP로 분류가 되어서 각 애플리케이션의 Recall이 낮았다. 또한 HANGAME과 SAYCLUB이 아닌 다른 애플리케이션들의 데이터 중 HTTP와 같은 특성을 가지는 많은 데이터들이 HANGAME과 SAYCLUB로 분류되어서 HANGAME과 SAYCLUB의 Precision이 낮은 것이다.

3.5.2 Byte 기반의 분류 결과 분석

그림 6은 4가지의 ML 알고리즘과 4가지의 Feature Set을 적용한 애플리케이션 트래픽 분류의 결과를 Byte 기반으로 측정한 정확도를 보여준다.

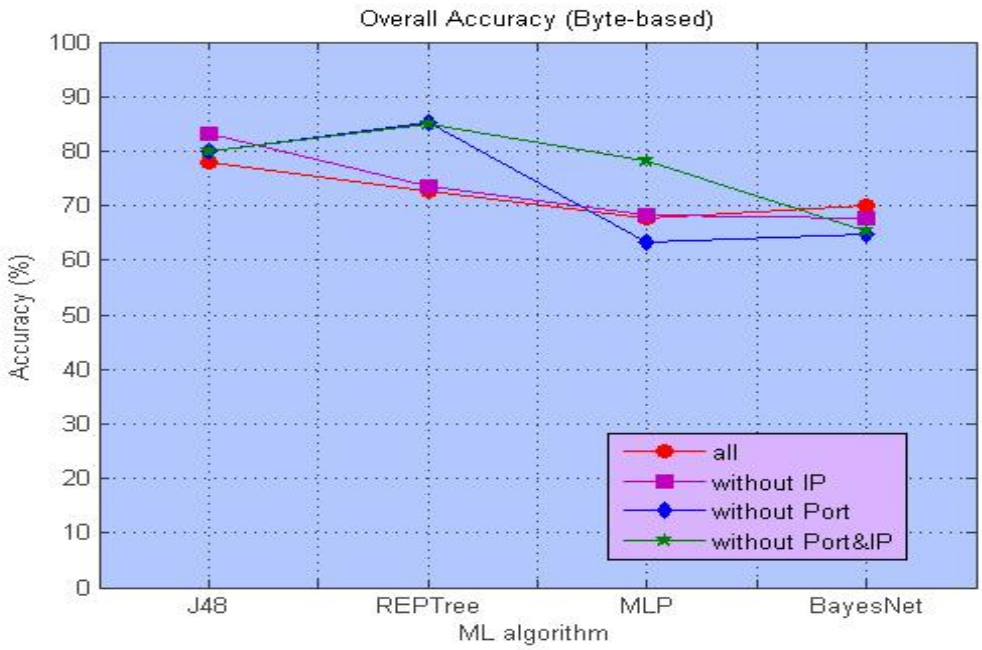


그림 6. Byte 기반 Overall Accuracy

J48 알고리즘을 4가지의 모든 Feature Set과 함께 적용하였을 때 약 82%의 정확도를 보였으며, REPTree와 Port를 제외한 Feature들로 구성된 (3)번 Feature Set을 적용하였을 때 약 86%의 가장 좋은 정확도를 보여준다. 반면에 Flow 기반에서 좋은 성능을 보인 BayesNet 알고리즘을 적용하였을 때에 약 70%의 좋지 못한 정확도를 보여준다. BayesNet 알고리즘을 적용한 애플리케이션 트래픽 분류 결과에서 Flow 기반으로 한 것이 Byte 기반으로 한 것 보다 높게 나온 것은 BayesNet 알고리즘이 많은 양의 트래픽을 발생하는 애플리케이션에 속한 트래픽을 제대로 분류하지 못했음을 의미한다. BayesNet 알고리즘을 적용한 애플리케이션 트래픽 분류 결과 중 Precision과 Recall을 보면 다른 알고리즘에 비해 FTP 프로토콜을 사용하는 애플리케이션에 대한 분류의 결과가 매우 좋지 않음을 알 수 있다. 즉, FTP 프로토콜을 사용하는 애플리케이션들은 큰 Byte 양을 가지는 적은 개수의

Flow를 만들므로 Flow 기반 분류 결과에는 큰 영향을 미치지 못하나, Byte 기반 분류 결과에는 소수의 Flow만 분류하지 못해도 Byte 양이 커서 Precision과 Recall의 값이 떨어진다.

표 8은 4가지 모든 Feature Set에 대해서 좋은 애플리케이션 트래픽 분류 결과를 보인 REPTree 알고리즘을 적용하고 3.4장에서 언급한 Feature Set들 중 Port를 제외한 (3)번 Feature Set을 사용하여 각 애플리케이션 별로 트래픽 분류를 수행한 결과를 보여준다. 이것은 Byte 기반으로 분석한 분류 결과로 Precision과 Recall 값을 나타낸다. Byte 기반 분석 결과도 Flow 기반의 결과와 같이 ALFTP와 PARANDISK, HANGAME, SAYCLUB, STARCRAFT의 분류 결과가 좋지 않다는 점이 비슷하다. ALFTP의 분류 결과는 Precision과 Recall은 거의 0에 가까웠고 PARANDISK의 분류 결과는 Precision은 약 13%이고 Recall은 약 31%로 Flow 관점으로 분석한 결과보다 좋지 않다. 이는 Flow 기반 분류에 적용한 Feature Set과 Byte 기반 분류에서 사용한 Feature Set이 각각 (1) Feature Set과 (3) Feature Set으로 다르기 때문이다. (3) Feature Set에는 Port 번호가 없기 때문에 Well-known Port 번호를 사용하는 FTP의 분류가 (1) Feature Set을 사용하는 것보다 좋지 않을 수 있다. FILEGURI의 경우 Flow를 기반으로 분석한 경우는 약 71%의 Precision과 55%의 Recall 값을 보인 것과 달리 Byte 기반으로 분석한 결과는 두 값이 각각 35%와 20%로 떨어지면서 분석을 제대로 하지 못했다. 이는 FILEGURI가 발생하는 Packet들 중 양(Byte)이 많은 트래픽이 GAMPLE과 SORIBADA에서 발생한 트래픽으로 잘못 분류되었기 때문이다. GAMPLE과 SORIBADA의 Precision이 Recall보다 낮은 것도 바로 이런 이유 때문이다.

Application	Precision	Recall
BITTORRENT	0.981	0.637
ALFTP	0.0045	0.0003
CLUBBOX	0.768	0.232
DESKTOP	1	1
FILEGURI	0.35	0.202
GAMPLE	0.953	0.981
HANGAME	0.43	0.102
HTTP	0.403	0.858
PARANDISK	0.133	0.311
SAYCLUB	0.003	0.02
SORIBADA	0.968	1
STARCRAFT	1	0.145
TOTODISK	0.97	0.09

표 8. Byte 기반의 애플리케이션 별 Precision & Recall

3.5.3 분석 결과

가장 높은 분류 정확도를 가지는 애플리케이션 트래픽 분류를 위한 ML 알고리즘과 Feature Set은 Flow 관점에서 보았을 때에 약 90%의 Overall Accuracy를 갖는 J48 알고리즘과 3.4장에서 정의한 모든 Feature들로 구성한 (1)번 Feature Set이고, Byte 관점에서 보았을 때에 약 86%의 Overall Accuracy를 갖는 REPTree 알고리즘과 3.4장에서 정의한 Feature들 중 IP와 Port 번호를 제외한 (4)번 Feature Set이다. 본 논문에서는 Flow 기반이나 Byte 기반에서 좋은 성능을 보이며 Feature Set에 따라 분류 결과의 편차가 작았던 J48 알고리즘을 본 논문에서 사용한 4가지 알고리즘 중 가장 좋은 알고리즘으로 제안한다. 그리고 Flow 기반이냐 Byte 기반이냐에 따라 3.4장에서 정의한 모든 Feature들로 구성한 (1)번 Feature Set과 IP와 Port 번호를 제외한 Feature들로 구

성한 (4)번 Feature Set이 애플리케이션 트래픽 분류에 가장 적합한 Feature Set이다.

또한, 3.5.1과 3.5.2에서 언급하였듯이, 표 4의 같은 카테고리 안의 각각의 애플리케이션 트래픽 분류를 하게 되면 비교적 좋지 못한 결과를 얻게 된다. 이 문제를 해결하기 위해서 각 애플리케이션의 독립적인 특징을 반영할 수 있도록 충분히 많은 Flow 데이터를 수집해야 할 필요가 있다. 본 논문은 [4]의 애플리케이션 트래픽 분류를 하기 위해 애플리케이션 당 Flow의 수가 25개 정도면 충분하다는 결론을 반영하여 데이터를 수집 하였다. 하지만 본 논문의 실험 결과로 미뤄보기에 애플리케이션 트래픽 분류를 하기 위해 한 애플리케이션 당 필요한 Flow의 수는 적어도 200개 이상이 되어야 할 것이다. 그리고, 각 애플리케이션의 독립적인 특징을 잘 나타낼 수 있는 새로운 Feature를 선정하는 것과 애플리케이션 트래픽 분류에 적합한 Feature Reduction 알고리즘을 적용하는 것도 필요하다.

4 성능 평가

이 장에서는 3.2장에서 언급한 수집 방법을 이용하여 호스트에서 수집한 데이터로 구성한 Training Set과 Internet Link에서 수집한 데이터로 구성한 Testing Set을 대상으로, 3.5장에서 ML 알고리즘 중 가장 좋은 애플리케이션 트래픽 분류 결과를 보인 J48 알고리즘을 적용하여 애플리케이션 트래픽 분류 결과와 NGMON을 적용하여 애플리케이션 트래픽 분류 결과를 비교 및 분석한다. 본 실험에서 수집한 데이터는 Ground Truth를 이용하여 분류의 정확도를 측정하기 위해 애플리케이션 트래픽 분류 대상의 모든 애플리케이션을 직접 발생시켜서 얻은 데이터다. 또한 Training Set과 Testing Set은 다른 시간에 수집한 데이터를 기반으로 구성하였다. ML 알고리즘 중 J48 알고리즘을 적용할 때 Training Set은 3.5장에서 사용한 Training Set과 거의 동일하며, 3.5장에서 분류 결과가 좋지 않은 애플리케이션 중 FTP와 PARANDISK의 많은 데이터를 수집하여 기존에 사용한 Training Set보다 더 나은 Training Set으로 모델을 Training 하였다. Testing Set은 3.5장에서 사용한 Testing Set과 동일한 것을 사용하였다.

4.1 NGMON과 ML 알고리즘을 적용한 분류의 결과

그림 7은 애플리케이션 트래픽 분류를 하기 위해서 ML 알고리즘 중 J48을 적용한 것과 NGMON을 적용한 것의 분류 정확도를 Byte 양을 기준으로 측정하여 각 애플리케이션 별로 보여준다. 애플리케이션 중 Overall이 의미하는 것은 모든 애플리케이션의 분류 정확도를 보여준다.

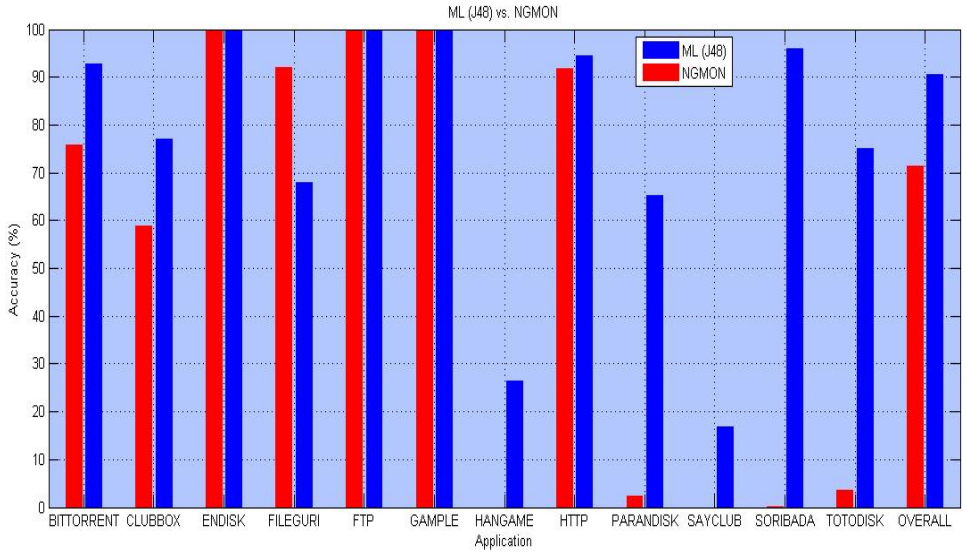


그림 7. ML 알고리즘(J48)과 NGMON의 분류 결과 비교

모든 애플리케이션을 나타내는 Overall을 살펴보면 J48 알고리즘을 적용한 분류 결과의 정확도는 약 90%를 보이고, NGMON을 적용한 분류 결과의 정확도는 약 71%를 보인다.

각 애플리케이션 별로 트래픽 분류의 정확도를 보면, BITTORRENT의 분류 정확도는 NGMON을 적용한 경우 약 75%이고 ML 알고리즘을 적용한 경우는 약 92%이다. CLUBBOX의 분류 정확도는 NGMON을 적용한 경우 약 59%이고, ML 알고리즘을 적용한 경우는 77%이다. ENDISK의 분류 정확도는 NGMON을 적용한 경우는 약 99%이고 ML 알고리즘을 적용한 경우는 약 99%이다. FILEGURI의 분류 정확도는 NGMON을 적용한 경우 약 99%이고 ML 알고리즘을 적용한 경우 약 67%가 나옴을 알 수 있다. FTP의 분류 정확도는 NGMON을 적용한 경우는 약 99%이고 ML 알고리즘을 적용한 경우는 99%이다. GAMPLE의 분류 정확도는 NGMON을 적용한 경우는 약 99%이고 ML 알고리즘을 적용한 경우도 약 99%이다. HANGAME의

분류 정확도는 NGMON을 적용한 경우는 0%에 가까웠고 ML 알고리즘을 적용한 경우 약 26%이다. HTTP의 분류 정확도는 NGMON을 적용한 경우는 91%이며 ML 알고리즘의 경우는 약 94%이다. PARANDISK의 분류 정확도는 NGMON을 적용한 경우는 2%에 가까웠으며, ML 알고리즘을 적용한 경우는 약 65%이다. SAYCLUB의 분류 정확도는 NGMON을 적용한 경우에 0%이며, ML 알고리즘을 적용한 경우 약 16%이다. SORIBADA의 분류 정확도는 NGMON을 적용한 경우는 약 0.3%이며, ML 알고리즘을 적용한 경우는 약 95%이다. TOTODISK의 분류 정확도는 NGMON을 적용한 경우는 약 3%이며, ML 알고리즘을 적용한 경우는 약 75%이다.

NGMON을 적용한 분류 결과와 ML 알고리즘을 적용한 분류 결과는 Web Browser을 사용하는 애플리케이션의 트래픽 분류의 정확도가 좋지 않다는 점은 비슷하다, 반면에 Web Disk나 P2P 애플리케이션의 경우, NGMON을 적용한 트래픽 분류의 정확도는 ML 알고리즘을 적용한 트래픽 분류의 정확도만큼 좋지 못하다. 특히 Port 번호를 80번을 사용하는 Web Disk 또는 P2P 애플리케이션의 경우 NGMON을 통한 트래픽 분류의 정확도는 5%를 넘지 않았다.

4.2 분류 결과의 분석

위의 2가지 분류 방법의 정확도가 다르게 나오는 이유는 다음과 같다. 첫째, NGMON을 적용한 애플리케이션 트래픽 분류에서 정의하는 애플리케이션의 의미와 ML 알고리즘을 적용한 애플리케이션 트래픽 분류에서 정의하는 애플리케이션의 의미는 약간 다르다는 점이다. 예를 들어 FTP와 같은 경우, NGMON에서의 FTP는 Port 번호로 20번이나 21번을 사용하는 것만을 말한다. 하지만 최근에 나오는 FTP 애플리케이션은 단순히 파일 전송하는 기능과 함께 FTP 애플리케이션을 사용하는 사용자들에게 광고를 전달한다거나 하는 Packet을 발생시키

는 부가적인 기능을 가지고 있다. ML 알고리즘을 적용한 애플리케이션 트래픽 분류에서는 이런 부가적인 기능으로 인해서 발생하는 Packet에 대해서도 FTP라고 정의한다. 본 논문에서는 FTP 애플리케이션 중 ALFPT 애플리케이션을 FTP라고 정의하였으며, 20번이나 21번을 사용하지 않는 Packet이더라도 이 애플리케이션이 FTP에 의해서 발생된 것이라고 판단되면 이것을 FTP 애플리케이션으로 분류가 되도록 정의하였다. 둘째, NGMON은 만들어 진지 약 3년 이상 된 것으로 그 이후로 달라진 애플리케이션의 변화에 대해서 반영이 미비한 상태이다. 따라서, FRM(Flow Relationship Map) 알고리즘에 관계없이 NGMON을 적용한 애플리케이션 트래픽 분류결과를 정확한 값이라고 판단하기 어렵다.

특히, P2P 애플리케이션인 BITTORRENT와 Port 번호를 80번으로 쓰는 애플리케이션인 SORIBADA와 TOTODISK 경우 NGMON을 적용하여 애플리케이션 트래픽 분류한 결과와 ML 알고리즘을 적용하여 애플리케이션 트래픽 분류한 결과의 정확도의 차이가 큰 이유를 각각 4.2.1과 4.2.2 에서 제시한다. 이로써 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 Port 번호를 80번으로 쓰는 애플리케이션이나 Dynamic한 Port 번호를 사용하는 P2P 애플리케이션이 존재하는 현재 네트워크 환경에 적합한 것임을 보인다.

4.2.1 BITTORRENT의 분류 결과 비교

NGMON을 적용한 애플리케이션 트래픽 분류 결과 중 BITTORRENT의 분류 정확도는 약 75%이다. 반면에 ML 알고리즘을 적용한 애플리케이션 트래픽 분류 결과 중 BITTORRENT의 분류 정확도는 약 92%이다. NGMON을 적용한 애플리케이션 트래픽 분류 결과가 비교적 낮은 이유는 BITTORRENT의 데이터를 HTTP로 분류를 하거나 또는 UNKNOWN으로 분류를 하기 때문이다. [19]는 FRM 알고리

즘을 이용한 NGMON이 BITTORRENT를 분류한 정확도가 Signature Matching을 적용하여 BITTORRENT를 분류한 정확도보다 낮음을 보여준다. 그림 8은 [19]에서 BITTORRENT를 Signature Matching을 적용한 분류 결과와 NGMON을 적용한 분류 결과를 비교한 것이다.

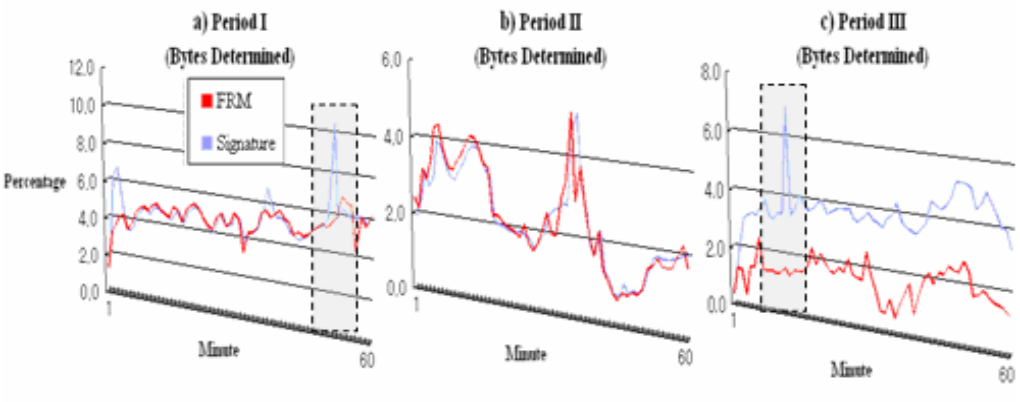


그림 8. Signature matching vs. FRM[19]

그림 8에서 보여주는 그래프를 보면 (a)와 (c)에서 NGMON을 적용하여 데이터를 BITTORRENT로 분류한 것보다 Signature을 적용하여 데이터를 BITTORRENT로 분류한 것이 약 2~3배 정도 많음을 알 수 있다. 이 결과가 의미하는 점은 NGMON을 적용하여 BITTORRENT를 분류하는 것은 적합하지 않다는 점을 알 수 있다. 또한 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 BITTORRENT를 분류하기에 더 좋은 방법임을 알 수 있다.

4.2.2 80번 Port를 사용하는 애플리케이션의 분류 결과 비교.

표 9는 Port 번호를 80번으로 쓰는 애플리케이션을 ML 알고리즘을 적용하여 애플리케이션 트래픽 분류한 결과와 NGMON을 적용하여 애플리케이션 트래픽 분류한 결과의 정확도를 보여준다..

Port 번호 80을 사용하는 Application	NGMON	ML(J48)
HTTP	91.6%	94.5%
GAMPLE	99%	99%
TOTODISK	3.642%	75%
PARANDISK	2.325%	65%
SORIBADA	0.3%	95%
HANGAME	0%	26%
SAYCLUB	0%	16.7%

표 9. 80번 Port 번호를 사용하는 애플리케이션

표 9의 애플리케이션은 Port 번호로 80번을 사용하는 애플리케이션이다. 표 9에서 애플리케이션 트래픽 분류의 정확도가 낮은 5개의 애플리케이션은 크게 두 가지로 나눌 수 있다. 먼저 HANGAME, SAYCLUB과 같은 Web Browser에서 동작하는 애플리케이션이 있다. 이 애플리케이션들은 HTTP로 분류 될 수도 있지만, HTTP 중 포털 사이트가 운영하는 애플리케이션으로 트래픽 분류가 가능하다. 다른 하나는 TOTODISK, PARANDISK, SORIBADA와 같이 두 개 이상의 서브넷에 속하는 다수개의 IP Address를 사용하는 Web Disk나 P2P 애플리케이션이 있다. 애플리케이션 트래픽 분류 결과를 위와 같이 두 가지로 나누어서 살펴보면, HTTP 애플리케이션 중 포털 사이트가 운영하는 애플리케이션의 경우에 NGMON을 적용하는 애플리케이션 트래픽 분류의 정확도와 ML 알고리즘을 적용하는 애플리케이션 트래픽 분류의 정확도, 모두 낮았다. 이런 결과를 갖는 이유는 HANGAME과 SAYCLUB 애플리케이션이 HTTP 애플리케이션과 다른 점이 많지 않아서 HTTP 애플리케이션으로 분류 되기 때문이다. 특히 NGMON을 적용하는 애플리케이션 트래픽 분류의 정확도가 거의 0에 가까운 이

유는 SAYCLUB, HANGAME 애플리케이션의 정의가 ML 알고리즘을 적용한 분류에서 이용한 것과 NGMON을 적용한 분류에서 이용한 것이 다르기 때문이라고 볼 수 있다.

TOTODISK, PARANDISK, SORIBADA와 같이 두 개 이상의 서브넷에 속하는 다수개의 IP Address를 사용하는 Web Disk나 P2P 애플리케이션이 분류 결과는 NGMON을 적용한 애플리케이션 트래픽 분류의 정확도에 비해, ML 알고리즘을 적용한 애플리케이션 트래픽 분류의 정확도는 높음을 볼 수 있다. 특히 SORIBADA의 경우가 가장 차이가 크다. 이는 NGMON을 적용하는 애플리케이션 트래픽 분류는 두 개 이상의 서브 넷에 속하는 다수개의 IP Address를 사용하는 Web Disk나 P2P 애플리케이션을 제대로 분류 할 수 없음을 의미한다. 반면에 ML 알고리즘을 적용하는 애플리케이션 트래픽 분류는 Web Disk나 P2P 애플리케이션 같이 두 개 이상의 서브 넷에 속하는 다수개의 IP Address를 사용하는 애플리케이션 트래픽 분류가 가능하다는 것을 보여준다. 이러한 결과는 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 Dynamic한 Port 번호를 사용하는 것과 같은 다양한 성격을 가지는 애플리케이션을 트래픽 분류하기에 더 적합하다는 점을 알 수 있다.

4.2.3 분석 결과

앞 장에서 ML 알고리즘과 NGMON을 적용하여 애플리케이션 트래픽 분류한 결과를 비교 및 분석함으로써 NGMON보다 ML 알고리즘이 실제 Internet Link의 데이터를 대상으로 애플리케이션 트래픽 분류에 더 적합하다는 점을 알 수 있다. 본 논문에서는 12개에 대한 애플리케이션에 대해서만 적용했다. 더 좋은 정확도를 가지는 애플리케이션 트래픽 분류를 하기 위해서 더 많은 애플리케이션에 대한 데이터를 수집하여서 Training Set을 얻어야 하며, 기존의 애플리케이션에

대한 데이터도 더 많이 수집해서 정확도를 높을 수 있도록 해야 할 것이다.

반면에 NGMON은 거의 100여 개의 애플리케이션에 적합하며 현재 실시간에 적합한 시스템을 구축해서 POSTECH의 Internet Link의 트래픽을 분류하고 있다. 현재의 애플리케이션의 변화에 대해서 반영한다면 더 좋은 정확도를 가지는 분류 결과를 가질 수 있을 것이다. 표 10은 분류 방법에 따른 장점과 단점 그리고 앞으로 나아가야 할 방향을 보여준다.

	장점	단점	앞으로 나아가야 할 방향
ML (J48)	• 다른 **Approach**에 비해 **Intelligent**한 기법을 적용하여 최근 애플리케이션 변화에 대응 • 패킷의 **Header**의 값만을 이용하여서 높은 정확도를 제공	• 단지 **12**개의 애플리케이션에 대해서만 적용	• 좀 더 많은 데이터를 이용하여 현재보다 나은 **Training Set**을 얻어야 함 • 좀 더 많은 애플리케이션에 적용할 수 있도록 개선해야 함 • 많은 애플리케이션에 대한 데이터 수집
NGMON (FRM)	• 많은 애플리케이션에 적합하게 대응 • 현재 **real time**에 적합한 시스템을 구축	• 구현하고 난 이후 최근 애플리케이션 변화를 적용하지 않음	• 최근 애플리케이션 변화를 적용하여야 함

표 10. 분류 방법에 따른 장, 단점 및 앞으로 나아가야 할 방향

5 결론 및 향후 과제

본 논문은 3.2장에서 언급한 수집 방법을 이용하여 얻은 데이터를 대상으로 다양한 ML 알고리즘과 Feature Set을 적용한 애플리케이션 트래픽 분류의 정확도를 Flow와 Byte의 측면에서 비교 분석하였다. 이를 통해, 본 논문에서 사용한 J48, REPTree, Multi-layer Perceptron, BayesNet의 4가지 ML 알고리즘 중 가장 성능이 좋은 ML 알고리즘으로 J48 알고리즘을 제시하였다. 또한, ML 알고리즘을 적용한 애플리케이션 트래픽 분류 결과와 NGMON을 적용한 애플리케이션 트래픽 분류 결과의 비교 및 분석을 통해서 ML 알고리즘을 적용한 애플리케이션 트래픽 분류가 Web Disk나 P2P 애플리케이션 같이 두 개 이상의 서브 넷에 속하는 다수개의 IP Address를 사용하는 애플리케이션이 존재하는 현재 애플리케이션의 변화에 대처할 수 있다는 것을 보였다.

본 논문에서 애플리케이션 트래픽 분류를 하기 위해 선정한 애플리케이션은 단일 Port 번호를 사용하는 애플리케이션뿐만 아니라, 2개 이상의 동적인 Port 번호를 사용하는 애플리케이션과, 기본 Port 번호로 80번을 사용하는 P2P 애플리케이션과 Web-Disk 애플리케이션도 다수 포함하였다. 특히 FTP 프로토콜과 P2P 프로토콜 그리고 HTTP 프로토콜을 사용하는 다수의 애플리케이션 분류를 시도함으로써 기존의 ML 알고리즘을 적용한 애플리케이션 트래픽 분류에서 보여주었던 결과인 Port 별 트래픽 분류가 아닌 실질적인 애플리케이션 별 트래픽 분류 결과를 보여준다.

또한, 더 나은 정확도를 갖고 실제 네트워크에 적용가능한 ML 알고리즘을 적용한 애플리케이션 트래픽 분류를 하기 위해 나아가야 할 방향은 좀 더 많은 데이터를 이용하여 현재보다 나은 Training Set을 얻어야 하며 많은 애플리케이션에 대한 데이터를 수집함으로써 좀 더 많은 애플리케이션에 적용 가능할 수 있도록 개선해야 합니다.

NGMON이 나아가야 할 방향은 최근 애플리케이션 변화를 반영하여서 분류의 정확도를 높일 수 있도록 해야 합니다.

앞으로는 같은 프로토콜을 사용하는 다양한 애플리케이션을 분류할 수 있도록 의미 있는 Feature들을 추출할 것이며, 기존의 애플리케이션을 포함한 좀 더 많은 애플리케이션에 대해 높은 정확도를 가지며 분류 할 수 있도록 더 많은 데이터를 수집할 것이다.

마지막으로 최소한의 Payload를 이용한 Signature를 적용한 분석 방법과 ML 알고리즘을 적용한 분석 방법을 함께 적용해서 애플리케이션 트래픽 분류를 한다면 좀 더 정확도가 높은 분류 결과를 얻을 수 있을 것이다.

참고문헌

[1] CAIA, http://caia.swin.edu.au/.

[2] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti, “Traffic Classification Using Clustering Algorithms”, SIGCOMM’06 Workshops, Pisa, Italy, Sep. 2006, pp. 281-286.

[3] Sebastian Zander, Thuy Nguyen, and Grenville Armitage, “Automated Traffic Classification and Application Identification using Machine Learning”, Proceedings of the IEEE Conference on Local Computer Networks, Sydney, Australia, Nov. 2005, pp. 250-257.

[4] Thuy T. T. Nguyen and Grenville Armitage, “Training on multiple Sub-Flows to optimize the use of Machine Learning classifiers in real-world IP Networks”, IEEE Conference on Local Computer Networks, Tampa, Florida, USA, Nov. 2006, pp. 369-376.

[5] Junghun Park, Hsiao-Rong Tyan, and C. C. Jay Kuo, “Inetnet Traffic Classification For Scalable QoS Provision”, IEEE International Conference on Multimedia and Expo, Jul. 2006, pp. 1221-1224.

[6] Andrew W. Moore and Denis Zuev, “Internet Traffic Classification Using Bayesian Analysis Techniques”, SIGMETRICS’05, Banff, Alberta, Canada, Jun. 2005, pp. 50-60.

[7] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt, “Internet Traffic Identification using Machine Learning”, IEEE Global Telecommunications Conference, California, USA. Nov.-Dec. 2006, pp. 1-6.

[8] N. Williams, S. Zander, and G. Armitage, “A Preliminary Performance Comparison of Five Machine Learning Algorithms for

Practical IP Traffic Flow Classification", SIGCOMM Computer Communication Review, Oct. 2006, pp. 7-15.

[9] Junghun Park, Hsiao-Rong Tyan, and C.-C. Jay Kuo, "GA-Based Internet Traffic Classification Technique for QoS Provisioning", International Conference on Intelligent Information Hiding and Multimedia, Pasadena, California, USA, Dec. 2006, pp. 251-254.

[10] Andrew Moore, Denis Zuev, and Michael Crogan, "Discriminators for use in Flow-based Classification", Technical Report, Intel Research Cambridge, 2005.

[11] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt, "Byte Me: A Case for Byte Accuracy in Traffic Classification", MineNet'07, J San Diego, California, USA, Jun. 2007, pp. 35-37.

[12] Machine Learning Lab in The University of Waikato, "Weka", [Online] Available: http://www.cs.waikato.ac.nz/ml.

[13] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju, and James W. Hong, "The Architecture of NG-MON: A Passive Network Monitoring System", IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, LNCS 2506, Montreal, Canada, Oct. 2002, pp. 16-27.

[14] Etheral, http://www.ethereal.com.

[15] N. Williams, S. Zander, and G. Armitage, "Evaluating Machine Learning Methods for Online Game Traffic Identification", CAIA Technical Report 060410C, Apr. 2006.

[16] M. Dash and H. Liu, "Consistency-based Search in Feature Selection", Artificial Intelligence, vol. 151, issue 1-2, 2003, pp. 155-176.

[17] M. Hall, "Correlation-based Feature Selection for Machine

Learning", PhD Diss. Department of Computer Science, Waikato University, Hamilton, NZ, 1998.

[18] Artificial Neural Network, http://en.wikipedia.org/wiki/Artificial_Neural_Network.

[19] Young J. Won, "A Hybrid Approach for Accurate Application Traffic Identification", MS Thesis, Dept. of Computer Sicence and Engineering, POSTECH, Feb. 2006.

[20] IANA, IANA Port number list, http://www.iana.org/assignments/Port-numbers.

[21] Andrew Moore, Denis Zuev, and Michael Crogan, "Discriminators for use in Flow-based Classification", Technical report, Intel Research Cambridge, 2005.

[22] S. Zander, T.T.T. Nguyen, and G. Armitage, "Self-learning IP Traffic Classification based on Statistical Flow Characteristics", Passive & Active Measurement Workshop(PAM) 2005, Boston, USA, Mar.-Apr. 2005, pp. 325-328.

[23] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin, Soule, and Kave Salamatian, "Traffic Classification On The Fly", ACM SIGCOMM Computer Communication Review Volume 36, Number 2, Apr. 2006, pp. 23-26.

[24] Augustin Soule, Kave Salamatian, Nina Taft, Richard Emilion, and Konstantina Papagiannaki, "Flow Classification by Histograms or How To Go on Safari in the Internet", SIGMETRICS/Performance'04, Jun. 12-16, 2004, New York, NY, USA, pp. 49-60.

[25] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark",

SIGCOMM'05, Aug. 21-26, 2005, Philadelphia, Pennsylvania, USA, pp. 229-240.

[26] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-Service Mapping for QoS: A statistical Signature-based Approach to IP Traffic Classification", ACM SIGCOMM Internet Measurement Workshop 2004, Taormina, Sicily, Italy, 2004, pp. 135-148.

[27] Ethem Alpaydin, "Introduction to Machine Learning", Published 2004 MIT Press ISBN 0262012111.

[28] Lei Yu and Huan Liu, "Feature Selection for high-dimensional Data: A fast correlation-based filter solution", In Proceedings of the Twentieth International Conference on Machine Learning, 2003, pp. 856-863.

[29] Martin Ester et al. "A Density-Based Algorithm For Discovering Clusters In Large Spatial Database With Noise", In International Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), Montreal, Canada, Aug. 1996, pp. 226-231.

[30] Peter Cheeseman and John Stutz, "Bayesian classification (AutoClass): theory and results", in Advances in Knowledge Discovery and Data Mining, U. M. Fayyad, G. Piatetsky – Shapiro, P. Smyth, and R. Uthurusamy, Eds., Menlo Park: The AAAI Press, 1995, pp. 153 - 180.

[31] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297.

[32] Autoclass, The Autoclass Project http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/.

[33] Arthur Dempster, Nan Laird, and Donald Rubin. "Maximum likelihood from incomplete data via the EM algorithm". Journal of the Royal Statistical Society, Series B, 1977, 39(1):1–38.

[34] Quinlan, J. R., “C4.5: Programs for Machine Learning”, Morgan Kaufmann, San Mateo, 1993.

[35] Y. Zhao and Y. Zhang, “Comparison of decision tree methods for finding active objects”, Advances in Space Research, Aug. 2007.

[36] T. Bayes, “An essay towards solving a problem in the doctrine of chances”, Philos. Trans. R. Soc. Lond. 53 (1763), pp. 370–418.

[37] Kevin B. Korb and Ann E. Nicholson, “Bayesian Artificial Intelligence”, CRC Press, 2004, ISBN 1584883871.

[38] Naïve Bayes Classifier, http://www.statsoft.com/textbook/stnaiveb.html.

[39] 한학용, “패턴인식 개론: MATLAB 실습을 통한 입체적 학습”, 한빛 미디어, 2006, ISBN 89-7914-323-0.

[40] Bayesian Network, http://en.wikipedia.org/wiki/Bayesian_Network.

[41] Decision Tree, http://www.autonlab.org/tutorials/dtree.html.