特別研究報告

題目

群知能を用いたクラスタリング手法の トラヒック分類への適用と評価

指導教員
村田 正幸 教授

報告者
須恵 匠

2014 年 2 月 14 日

大阪大学 基礎工学部 情報科学科

平成 25 年度 特別研究報告

群知能を用いたクラスタリング手法の
トラヒック分類への適用と評価

須恵 匠

内容梗概

インターネットの普及に伴い、多種多様なサービスがインターネットを介して提供されるようになり、ネットワークに対するサービスの性能要求も多岐にわたる状況となってきている。ネットワーク管理者はサービスによって異なる性能要求を満たしつつ、各サービスのトラヒックを収容する必要がある。このように多様な性能要求に合わせてトラヒックの収容を行うためには、ネットワーク内を流れるトラヒックを性能要求に合わせて分類することが必要となる。

性能要求に応じてトラヒックの分類・識別を行う手法のとして、トラヒックの統計情報をもとにした手法が注目されている。これらの手法では、各フローの統計情報を取得し、機械学習を用いて各フローの分類を行う。機械学習手法の中でも、特にクラスタリングと呼ばれる類似のデータを集めたクラスタを構成する手法は、事前に各分類先の特徴を学習することなく分類を行うことが可能であり広く用いられている。

近年、データマイニングの分野においてクラスタリング手法に関する研究も進められており、中でも生物学に知見を得た手法はクラスタリングの正確性と計算量のバランスの良さから注目を集めている。これらの新たなクラスタリング手法をトラヒック分類に適用することにより、正確な分類と短い計算時間での分類の両立が可能となると考えられる。しかしながら、これらの新たな手法をトラヒック分類へ適用した際の性能については、十分な議論が行われていない。

本報告では、 Web アプリケーションによるトラヒックの分類に焦点をあて、トラヒック分類器に群知能にもとづいた手法を含む複数のクラスタリング手法を適用し、それらの比較評価を行う。その結果、従来型のクラスタリング手法である K-means 法や群平均法は、パラメータを分類対象のデータ内に含まれているトラヒックの種類の数に合わせて設定しないと、6 割以上のフローを誤ったクラスタに分類ししまうアプリケーションが生じるなど、正確な分類ができないのに対して、群知能にもとづいたクラスタリング手法である AntTree は、単一のパラメータ設定により分類対象のデータ内に含まれているトラヒックの種類の数

によらず、すべてのアプリケーションに対して 6 割以上の割合でフローを正しいクラスタに分類できることが分かった。

主な用語

トラヒック分類、クラスタリング、群知能、 Web アプリケーション

目 次

図 目 次

表 目 次

1　はじめに

インターネットの普及に伴い多種多様なサービスがインターネットを介して提供されるようになり、サービスのネットワークに対する性能要求も多岐にわたるようになってきた。たとえば、動画ストリーミングサービスでは、生成された動画のビットレートに応じた通信帯域を安定して確保可能なネットワークが望まれる一方、ゲームなどのインタラクティブな操作を必要とするアプリケーションでは即応性が重要であり、サービス提供者とユーザ間の低遅延な通信が必要とされる。

ネットワーク管理者はアプリケーションによって異なる性能要求を満たしつつ、各アプリケーションのトラヒックを収容する必要がある。そのため、異なる性能要求を持つサービスを一つのネットワーク上に収容する手法の検討が進められており、たとえば文献 [1] では、サービスごとに仮想ネットワークを構築し、各サービスの要求にあわせて各仮想ネットワークを動的に制御しつつ仮想ネットワーク間の資源を調停する手法が提案されている。このようなアプリケーションの性能要求に合わせたネットワーク制御を行うためには、ネットワーク内を流れるトラヒックをアプリケーションに合わせて分類することが必要となる。

従来、トラヒックのアプリケーション識別は、 TCP・UDP のポート番号をもとに行われてきており [2]、たとえば 25 番のポートを用いて行われている通信はメールの送信を行っていると判断され、 80 番のポートを用いて行われている通信は、 Web ページのブラウジングと判断されてきた。しかしながら近年、 YouTube などの動画共有サービスからゲームなどのインタラクティブなアプリケーションまで、多種多様なサービスが Web ブラウザを介して提供されるようになってきた。その結果、ネットワークへの性能要求の異なる多種多様なサービスが HTTP プロトコルを用いて提供されるようになってきており [3]、それらはすべて 80 番ポートや 443 番ポートを用いて通信を行うため、ポート番号によるトラヒックの種別の分類は困難となってきている。

ポート番号によらずトラヒック識別を行う手法のとして、トラヒックの統計情報をもとにした手法が注目されている [2, 4-6]。パケットサイズやパケットの到着間隔の平均や分散、分布はアプリケーションや通信に用いられるプロトコルによって大きく異なることが知られている。統計情報を用いた手法ではこれを利用し、各フローについてパケットのサイズや到着間隔などの統計情報を取得し、その統計情報をもとにフローの分類を行う。統計情報取得後、各フローの分類には機械学習が用いられる。特に、クラスタリングと呼ばれる類似のデータを集めたクラスタ (塊) を構成する手法は、事前に各分類先の特徴を学習することなく分類を行うことが可能であり広く用いられている [4, 6]。統計情報を用いたフローの分類では分類に用いる機械学習手法が大きな影響を与え、同じ統計データを用いた場合であっても用いた機械学習手法によって精度が大きく異なり、また、分類にかかる速度も機械学習手

法によって大きく異なる。トラヒック分類に用いる機械学習手法に関する検討も行われているものの [6]、通信に用いられているアプリケーションプロトコルの識別に焦点をあてた手法の検討がほとんどであり、Web アプリケーションに関するトラヒックについて、そのネットワークへの性能要求に応じた分類を行うことを目的とした分類に焦点をあてた検討は十分に行われていない。

さらに近年、データマイニングの分野においてクラスタリング手法に関する研究も進められており、中でも生物学に知見を得た手法は計算量とクラスタリングの正確性のバランスの良さから注目を集めている [7]。これらの新たなクラスタリング手法をトラヒック分類に適用することにより、分類の正確性と短い計算時間での分類の両立が可能となると考えられる。しかしながら、これらの新たな手法をトラヒック分類へ適用した際の性能について十分な議論は行われていない。

本報告では、 Web アプリケーションによるトラヒックの分類に焦点を当て、トラヒック分類機に生物学の知見にもとづいた分類手法を含む複数のクラスタリング手法を適用し、それらの比較評価を行う。比較にあたり、 Web アプリケーションを利用している際に発生したパケットをキャプチャし、キャプチャしたパケットトレースデータから分類に用いるトラヒック統計情報を取得した。その後、取得したトラヒック統計情報をもとに各クラスタリング手法を動作させ、その正確性と計算時間について比較を行うとともに、ネットワークへの性能要求に応じた分類を行うという観点から、各クラスタリング手法の有用性について考察する。

以降、2 章において関連研究について述べ、3 章でクラスタリング手法について紹介する。その後、本報告で用いたクラスタリング手法を適用したトラヒック分類手法について 4 章で述べ、5 章において、それらの比較評価を行い、各手法の有用性について考察を行う。最後に 6 章で、まとめと今後の課題について述べる。

2 関連研究

2.1 ポート番号に基づくトラヒック分類

TCP/IP による通信では、 IP アドレスにより通信相手を指定し、ポート番号により通信相手内のプログラムを特定して通信を行う。ネットワークを介してサービスを提供しているコンピュータでは、サーバープログラムは、クライアントからの接続要求を待ち受けるポート番号を指定して起動される。一般的なネットワークアプリケーションについては、Internet Assigned Numbers Authority (IANA) が定めた Well-known Port と呼ばれるポート番号を指定してサーバープログラムが起動されることが多い。そのため、通信パケットのヘッダをもとに、宛先ポート番号あるいは送信元ポート番号に Well-known Port が含まれていれば、当該アプリケーションのパケットであると識別することが可能である。ポート番号によるアプリケーション種別の識別は広く用いられており、一般的な Firewall でもポート番号によって通過させるトラヒック・遮断させるトラヒックを決定している。

しかしながら、近年、 YouTube などの動画共有サービスからゲームなどのインタラクティブなアプリケーションまで、多種多様なサービスが Web ブラウザを介して提供されるようになってきた。その結果、ネットワークへの性能要求の異なる多種多様なサービスが HTTP プロトコルを用いて提供されるようになってきており [3]、それらはすべて 80 番ポートや 443 番ポートを用いて通信を行う。ポート番号による分類では、それらのトラヒックは HTTP あるいは HTTPS と判別されるのみで、 HTTP を介して提供される性能要求の異なる多様なサービスを識別することはできない。

2.2 シグネチャに基づくトラヒック分類

ポート番号によらずにトラヒックの分類を行う手法の一つが、パケットのヘッダのみではなくペイロードを分析する手法である [2, 8, 9]。これらの手法では、識別したいアプリケーションのパケットが持つ特徴的なビットパターンをあらかじめ定義しておき、パケットのペイロード内やペイロードから再構築された通信データ内に定義されたパターンを含んでいるかを調べることにより、アプリケーションの識別を行う。

ペイロードを用いたトラヒックの識別では、特定のアプリケーションやセキュリティ上問題のある通信を検出することに焦点をあてた研究が進められている。しかしながらこの方法は事前に定義されたアプリケーションの通信を判別することしかできず、パターンの定義がされていない新たなアプリケーションの分類を考慮したものではない。

2.3 統計情報に基づく識別

パケットサイズやパケットの到着間隔の平均や分散、分布はアプリケーションや通信に用いられるプロトコルによって異なることが知られている。これを利用し、トラヒック統計情報を用いてトラヒックを分類・識別する手法が検討されている [2, 4-6]。文献 [4, 6] では、パケットのサイズ、到着間隔の平均・分散の情報を用いて、トラヒックの識別を行う手法が提案されている。この手法では、識別先が既知のトラヒックの到着間隔の平均・分散の情報とその分類先の情報を教師データとして用い、トラヒックの分類方法の学習を行う。そして学習された分類方法をもとに、新たに到着したトラヒックの分類を行う。文献 [10] では、通信初期のパケットサイズの時系列データをもとにトラヒック識別を行う手法が提案されている。この手法は、通信初期のパケットサイズの時系列データにアプリケーションの特徴が現れるということを利用したものであり、識別先が既知であるトラヒックのパケットサイズの時系列データを教師データとして用い分類ルールを学習する。そして、新たに到着したトラヒックの通信初期のパケットサイズの時系列データに対して学習データを適用し、トラヒックの分類を行う。この手法では、各フローについて通信開始初期にトラヒックの分類を行うことができる。これらの教師データをもとにして分類ルールを学習する手法では、教師データに含まれているものと近い特徴をもつトラヒックは正確に分類できるものの、教師データに含まれていない種類のトラヒックの分類を行うためには新たな教師データを用いた再学習が必要となる。

そこで、教師なしの機械学習であるクラスタリング手法を用いたトラヒック分類手法についても検討が進められている [11, 12]。クラスタリングは、入力として与えられたデータの集合のうち似ているデータをまとめたクラスタを生成する手法であり、トラヒック統計情報に対してクラスタリング手法を適用することにより、統計的特徴の近いトラヒックのグループの情報を得ることができる。クラスタリング手法を用いたフローの分類では、分類に用いるクラスタリング手法が大きな影響を与え、同じ統計データを用いた場合であっても、用いたクラスタリング手法によって生成されるクラスタが異なり、また分類にかかる時間も異なる。複数のクラスタリング手法をトラヒック分類に用いた場合の性能比較も行われているものの [11]、通信に用いられているアプリケーションプロトコルの識別に焦点をあてた手法の検討がほとんどであり、Web アプリケーションに関するトラヒックについて、そのネットワークへの性能要求に応じた分類を行うことを目的とした分類に焦点をあてた検討は十分に行われていない。

さらに近年、データマイニングの分野においてクラスタリング手法自体に関する研究も進められており、中でも生物学に知見を得た手法は計算量とクラスタリングの正確性のバランスの良さから注目を集めている [7]。これらの新たなクラスタリング手法をトラヒック分類

に適用することにより、正確な分類と短い計算時間での分類の両立が可能となると考えられる。しかしながら、これらの新たな手法をトラヒック分類へ適用した際の性能については十分な議論が行われていない。

3　クラスタリング (クラスター分析)

3.1　クラスタリングの概要

クラスタリングはデータの集合を入力とし、入力されたデータを類似するデータを集めた複数のクラスタに分割する手法である。各データは N 個の値を持ち (N 次元) 、N 個の値を用いてデータ間の類似度が定義される。そしてクラスタリング手法では、それぞれのクラスタに類似度の高いデータが集まるようにクラスタが形成される。

様々なデータ間の類似度の定義が存在するが、その中でもよく使われるのがユークリッド距離である。データ x とデータ y のユークリッド距離は以下で定義される。

$$Dist(x,y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{1}$$

ただし、x_i はデータ x の i 番目の値を示す。以降の本報告においてもクラスタリングを行う際には、類似度の定義として基本的にユークリッド距離を用いる。

3.2　従来型クラスタリング手法

3.2.1　K-means 法

クラスタリング手法で最も広く使われている手法の一つが K-means 法である。K-means 法はクラスタ数 k を入力パラメータとし、以下の手順により入力されたデータを k 個のクラスタに分割する。

1. 各データを k 個のクラスタにランダムに分割する
2. 割り振られたデータをもとにクラスタ内のデータの算術平均を計算し、得られた値をクラスタの中心と定義する
3. 各データについて、各クラスタの中心との距離を計算し、当該データをもっとも中心と距離が近いクラスタに移動する
4. データの移動が発生しなかった場合は処理を終了、データの移動が発生した場合は手順 2 に戻る

K-means 法の結果は、手順 1 で行う各データのランダムなクラスタへの割り当てに依存する。そのため、よりよい結果を得るために初期値の与え方を変えた K-means++法 [13] が提案されている。

K means++法は、上記の K-means 法の手順 1 の変わりに以下の手順を行い、初期のクラスタの中心を計算する。

1. データからランダムに 1 つ選びそれをクラスタの中心とする
2. それぞれのデータ x 対して、各クラスタの中心との距離を計算し、その距離の最小値を D_x とする
3. 各データ点 x に対して、D_x^2 に比例する重み付きの確率分布を用いて、新たなデータを選択し、選択したデータをクラスタの中心とする
4. 手順 2、3 を繰り返し、k 個のクラスタの中心を選択する。

k 個のクラスタの中心が選択された後は、通常の K-means 法の手順によりクラスタリングを行う。

本報告では、より正確なクラスタリングが可能な K-means++法を評価対象のクラスタリング手法の一つとして用いる。

3.2.2 群平均法

各データそれぞれが 1 つのデータを含むクラスタである、とみなした状態から再帰的に類似したクラスタを統合することにより、より大きなクラスタを構成する手法である。群平均法では、クラスタ C_1、C_2 間の距離を以下のように定義する。

$$Dist(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{x_1 \in C_1} \sum x_2 \in C_2 Dist(x, y) \tag{2}$$

このように定義することにより、クラスタ C_1、C_2 内の全要素が類似している場合はクラスタ間の距離が短くなり、ひとつでも類似していない要素があればクラスタ間の距離が大きくなる。

群平均法では、クラスタの統合を繰り返すことにより階層的にクラスタが形成される。本報告の評価では、K 個のクラスタができた時点でクラスタリングの処理を終了するものとした。

3.3 群知能に基づくクラスタリング手法

群知能は動物の群れの行動に着想を得た計算手法であり、近年多くの手法が研究されている。クラスタリングの分野でも群知能を用いた手法の検討が進められている。

3.3.1 AntTree

群知能に基づく代表的なクラスタリング手法がアリの行動を模倣した AntTree [7, 14] である。この手法はアリが互いに寄り集まって鎖状につながり、複雑な構造を形成する行動をもとにしたものである。アリはスタート地点 (サポート) から出発する。一部のアリがサポートとつながり始め、つながったアリにさらに別のアリが繋がる。繋がったアリは組織の一部となり、他のアリはその組織を伝って移動するようになり、さらに別のアリとつながる。この手順が繰り返されることにより、複雑で大規模な組織が構築される。

AntTree ではこのアリの行動様式を模して各データをアリとみなし、アリの行動様式に従ってデータが他のデータとつながる。本手法では、データ同士がつながる際に類似のデータのみがつながるようにすることにより、類似のデータがつながった鎖を構築する。

各アリ (データ) は、二つの閾値 T_S と T_D を持つ。また、各アリは他の一匹のアリのみと繋がることができ、現在鎖状に組織を構成したアリの上を移動しつつ、どのアリと繋がるのかを以下のルールに従って自律的に判断する。

- 現在地がサポートである場合
  - サポートと繋がったアリがいない場合はサポートと繋がる
  - サポートと繋がったアリがいる場合は
    - サポートにつながったアリのうち、もっとも自身と似たアリを探す
    - もっとも自身と似たアリとの類似度が T_S よりも大きければ、そのアリに移動する
    - もっとも自身と似たアリとの類似度が T_D よりも小さければ、サポートにつながる
    - それ以外は、$T_S \leftarrow \alpha_1 T_S$、$T_D \leftarrow T_D + \alpha_2$ として閾値を更新する
- 現在地がサポート以外であれば、
  - 現在地のアリとの類似度が T_S よりも大きければ
    - 現在地のアリと繋がっているアリのうち、もっとも自身と似たアリを探す
    - もっとも自身と似たアリの類似度が T_D よりも小さければ、現在地のアリと繋がる
    - もっとも自身と似たアリの類似度が T_D よりも大きければ、$T_S \leftarrow \alpha_1 T_S$、$T_D \leftarrow T_D + \alpha_2$ として閾値を更新した上で、ランダムに選択した現在地のアリと繋がっているアリに移動する

– 現在地のアリとの類似度が T_S 以下であれば、ランダムに選択した現在地のありと繋がっているアリに移動する

上記のアルゴリズムにおいて T_S、T_D の初期値、α_1、α_2 はクラスタの形成のされやすさを決めるパラメータである。以後の本報告の評価では、T_S の初期値は 1、T_D の初期値は 0 とし、α_1、α_2 を変化させた複数のパターンについて評価を行った。また、AntTree では類似度は 0 から 1 の間で表現される値であるため、xy 間の類似度 $Sim(x, y)$ を $Dist(x, y)$ と $Dist_{max}$ を用いて、以下のように定義した。

$$Sim(x, y) = 1 - \frac{Dist(x, y)}{Dist_{max}} \tag{3}$$

$$Dist_{max} = max(Dist(\forall x, \forall y)) \tag{4}$$

そして、この手順により、全アリがいずれかのアリにつながった時点でクラスタリングを終了する。その時点で、サポートにつながっているアリの部分木をそれぞれのクラスタとみなす。

4　クラスタリング手法のトラヒック分類への適用

4.1　概要

本報告でのトラヒックの分類は分類器において行われる。分類器はネットワークのアクセスルータに設置され、アクセスルータを流れるトラヒックを観測する。観測は、同一送信元 IP アドレス・宛先 IP アドレス間の一連の通信をフローとして定義し、フロー単位で行う。フローは当該 IP アドレス間の最初のパケットが到達した時点で開始したとみなされ、一定期間当該 IP アドレス間の通信が行われなかった場合にフローは修了したとみなされる。各フローに対する統計情報はフロー終了検出時に集計され、クラスタリング手法の入力として使われる。そしてクラスタリング手法を用いて各フローの分類を行う。

クラスタリング手法を適用することにより、各フローは類似したフローを集めたクラスタに分けられる。その後、クラスタ内のフローが関係するアプリケーションにもとづき各クラスタにラベルを付けることにより、各フローの分類を完了する。ラベルはクラスタリングされた結果にもとづき手動でつけることもできれば、クラスタ内に既知のアプリケーションフローが存在した場合は、そのアプリケーションに関連するラベルを自動で付与することも可能である。

本報告ではラベルの付与方法については扱わず、5 章の評価では、クラスタリング完了時にアプリケーションに応じたクラスタが構成されているのかについて評価を行う。

4.2　分類に用いる統計情報

本報告では、 Web アプリケーショントラヒックの分類を目標としている。そこで本節では、 Web アプリケーションで発生するトラヒックの性質について説明し、その後分類に用いる統計情報について説明する。

4.2.1　Web アプリケーションで発生するトラヒック

Web アプリケーションによる通信は、すべて HTTP を用いて行われる。HTTP は次の手順により通信が行われる。まず、クライアントからサーバに対して TCP の接続要求を送り接続を確立する。接続を確立した後は、クライアントからデータの取得を要求する GET や、データの送信を行う POST リクエストを行う。サーバ側はクライアントからの要求に応じて、レスポンスとしてクライアントが要求したデータを返答する。

Web アプリケーションによる通信は、アプリケーションの種類によらず上記の手順に従う。しかしながら、アプリケーションによってクライアントが GET や POST のリクエス

トを行うタイミングや、各リクエストに対するレスポンスの大きさが異なる。そこで本報告では、そのような特徴を捉えることができるようなトラヒック統計情報を用いてアプリケーションの識別を行う。実際に本報告で用いたトラヒック統計情報については、4.2.2 節で述べる。

4.2.2 Web アプリケーションの識別に用いるトラヒック統計情報

本報告では、 Web アプリケーションの挙動に起因する以下の指標を用いてクラスタリングを行う。

上下方向のペイロード付パケット数の比 本報告では、 Well-known Port 宛の通信を上り、Well-known Port からの通信を下りと定義する。そして、上り方向のペイロード付パケット数 P^{UP} と、下り方向のペイロード付パケット数 P^{DOWN} をカウントし、以下の式で定義される値を計算する。

$$\log \frac{P^{DOWN}}{P^{UP}} \tag{5}$$

ペイロード付パケットのみをカウントすることにより、 ACK のみのパケットは除外され、上り方向のクライアントからのリクエスト関係するパケット数、下り方向のサーバらからのレスポンスに該当するパケット数のみを数えることができる。そのため、式 (5) の値は、クライアントからのリクエストとサーバからのレスポンスに該当するパケットの比率となる。この値が大きければ、少ないクライアントのリクエストに対して多量のレスポンスが返されているとみなすことができる。その一方、この値が小さければ、クライアントは多くのリクエストをサーバに対して送信しており、インタラクティブな通信や、クライアントからのデータのアップロードが行われているとみなすことができる。

下り方向のペイロード付パケット到着レートの変動係数 同一サーバから同一クライアントへ送られているトラヒックのうち、ペイロード付パケットの到着レートを計測する。本報告では、パケットの到着レートは各タイムスロットあたりに到着したパケットの数と定義し、5 章の評価の際には、タイムスロットの長さを 5 秒とした。R_i^{DOWN} を i 番目の下り方向のペイロード付パケットの到着レートとし、該当フローが通信を行っている時間が M 個のタイムスロットに分割できるとすると、当該フローの平均パケット到着レートは以下のように定義される。

$$Avg^{DOWN} = \frac{1}{M} \sum_{i=0}^{M} R_i^{DOWN} \tag{6}$$

また、当該フローの下り方向ペイロード付パケットの到着レートの標準偏差は以下のように定義される。

$$Std^{DOWN} = \sqrt{\frac{1}{M-1}\sum_{i=0}^{M}(R_i^{DOWN} - Avg^{DOWN})} \tag{7}$$

変動係数は、Avg^{DOWN} と Std^{DOWN} を用いて以下のように定義される。

$$CV^{DOWN} = \frac{Std^{DOWN}}{Avg^{DOWN}} \tag{8}$$

CV^{DOWN} はサーバからのレスポンスパケットの到着レートのばらつきを表す。CV^{DOWN} が小さければ、一定のレートでレスポンスパケットが到着し続けていることを示し、ストリーミング系のアプリケーションなど、下り方向で一定のレートでの通信を行い続けるフローであると判断することができる。

上り方向のペイロード付パケット到着レートの変動係数　同一クライアントから同一サーバへ送られているトラヒックのうち、ペイロード付パケットの到着レートを計測する。そして下り方向のペイロード付パケットの到着レートと同様、その到着レートの平均、標準偏差をもとに変動係数を以下のように計算する。

$$Avg^{UP} = \frac{1}{M}\sum_{i=0}^{M} R_i^{UP} \tag{9}$$

$$Std^{UP} = \sqrt{\frac{1}{M-1}\sum_{i=0}^{M}(R_i^{UP} - Avg^{UP})} \tag{10}$$

$$CV^{UP} = \frac{Std^{UP}}{Avg^{UP}} \tag{11}$$

CV^{UP} はクライアントからのリクエストパケットの到着レートのばらつきを表す。CV^{UP} が小さければ、一定のレートでクライアントからリクエストが送られていることを示す。ストリーミング系のアプリケーションでは、ストリーミングのデータは小さなデータに分割され、クライアントはサーバに対して次のデータに対するリクエストを定期的に送信する。CV^{UP} を用いることにより、そのような定期的にサーバに対してリクエストを送っているようなフローを検出することが可能となる。

5 評価

5.1 評価方法

5.1.1 分類対象のトラヒックデータ

本報告では、大阪大学内の PC からブラウザ (Internet Explorer 11) を用いて Web アプリケーションを利用し、その際に発生したパケットを当該 PC 上で動作させたネットワークアナライザソフトウェア Wireshark[15] を用いてキャプチャしたデータを用いる。

本報告では、ネットワークへの要求の異なる以下の 5 種類の Web アプリケーションを利用した際のパケットキャプチャデータを用いた。パケットキャプチャは、各種類の Web アプリケーションあたり 10 回行った。

分類結果はネットワーク制御に用いられることを想定しており、ネットワーク制御への影響が大きいフローの分類に焦点をあてた評価を行う。そのため、フロー内に含まれるデータが 5 MByte 以下のフローは小規模フローと分類されるものとし、5 MByte 以上の大規模フローを分類対象とした。

また、分類器の設置個所により、観測されるアプリケーションの種類には偏りが生じると考えられる。この偏りが生じた場合の分類性能についても評価するため、以降の本評価では以下の 5 種類の Web アプリケーションのトラヒックのうち、3 種類が分類対象のトラヒックデータに含まれている場合、すべてのアプリケーションのトラヒックデータ含まれている場合の評価を行った。

Download ユーザが所望のファイルをダウンロードする Web アプリケーションであり、クライアント側からのリクエストに対して、サーバはリクエストに指定されたファイルの送信を行う。当該アプリケーションでは、ユーザが高速に必要なファイルを取得することが望まれるため、サーバからクライアントへの十分に大きな通信帯域が確保されることが望ましい。一方、上り方向はファイル取得のリクエストしか送信されないため、大きな帯域の確保は不要である。本報告では、Bitcasa[16]、 Box[17]、 Dropbox[18] の 3 種類のクラウドサービスからファイルをダウンロードした場合のパケットキャプチャデータを用いた。

Upload ユーザがクラウドサービス等にファイルをアップロードし、別の端末や他のユーザとの共有を行うアプリケーションである。ファイルのアップロードを行う際には、クライアントからサーバに多量のトラヒックが発生する一方、サーバからクライアントへ送られるトラヒック量は少ない。そのため、クライアントからサーバ方向の通信に対して十分に大きい帯域を確保することが望まれる。本報告では、Download と同じく、Bitcasa[16]、

Box[17]、 Dropbox[18] の 3 種類のクラウドサービスに対してファイルをアップロードした際にキャプチャを行ったデータを用いた。

Video Live Streaming　イベントを撮影した動画をリアルタイムで配信するサービスである。クライアント側は定期的に次の時刻の動画に対するデータを要求するリクエストをサーバ側に送信し、サーバはクライアントの要求に合わせてクライアントに新たに撮影された動画を送信するという形で、Web アプリケーションとして動画のストリーミングサービスが実現されている。本アプリケーションでは、各時刻で撮影された動画のデータがサーバからクライアントに送信され、クライアントからサーバへは新たな時刻の動画データの要求が定期的に送信される。本サービスは、ネットワークに対しては下り方向で各時刻の動画データを送信するだけの帯域が安定的に提供されることを要求する。本報告では、Ustream[19] の生放送を視聴した際に発生したパケットをキャプチャしたデータを用いる。

Video on Demand　ユーザが所望した動画ファイルをダウンロードして再生を行うアプリケーションである。本アプリケーションも、Video Live Streaming と同様ひとつの動画は複数のデータに分割され、その分割された各データに対してクライアントが順にリクエストを送ることにより必要な動画像データを受信しながら再生を行うという形で、Web アプリケーションとして実現されている。Video on Demand は、Video Live Streaming とは異なり現在撮影された動画像ではなく、サーバ側に蓄えられている動画データが送信される。クライアント側では、サーバから送られてきた動画像データをキャッシュしながら再生を行う。そのため、十分な帯域が存在していれば送信可能な分の動画像データがクライアントに送られる。本報告では、YouTube[20]、Ustream[19] の録画放送配信 、ニコニコ動画 [21] を利用した際のパケットをキャプチャしたデータを用いた。

Interactive　ユーザの操作に応じてサーバ側にリクエストが送られ、ユーザの操作に応じた返答がサーバから行われるようなアプリケーションである。本報告では、このようなアプリケーションとして地図情報サービスを用いた。地図情報サービスでは、ユーザがクライアント端末側で表示された地図を動かした際には、動かした先の新しい地図情報をサーバ側から取得して表示を行う。このアプリケーションではユーザが操作したタイミングでサーバ側にリクエストが送られ、リクエストされた情報をサーバが返答する。このようなユーザの操作に応じてサービスを提供するアプリケーションでは即応性が重要であり、低遅延の通信が必要となる。本報告では、Google マップ [22]、 Yahoo!地図 [23] を利用した際にキャプチャしたデータを用いた。

5.1.2 評価指標

クラスタリング結果を評価するにあたり、以下の 2 つの指標を用いた。

クラスタリングの正確性 Web アプリケーションの分類結果がネットワーク制御の入力として用いることを考えると、各フローが自身のアプリケーションとはネットワークに対する性能要求が異なるフローと同一のクラスタに分類されることは避ける必要がある。そこで、本評価では、各フローが自身のアプリケーションと同じ性能要求を持つクラスタに正しく分類されているかを示す正確性に関する指標を定義し、比較を行う。

正確性を定義するにあたり、まず、各クラスタに以下の条件のいずれも満たすアプリケーションのラベルを付与する。

- 当該クラスタに属するアプリケーション種別のうち最多もの
- 当該クラスタに 3 つ以上の当該アプリケーション種別のフローが含まれている

上記の条件を満たすアプリケーションが存在しない場合、当該クラスタは適切なラベルを付けることができないクラスタであるとする。

その後、各フローについて当該フローのアプリケーション種別と同じラベルが付与されたクラスタに分類された割合を正確度として定義する。アプリケーション種別 i の正確度 C は以下のように定義される。

$$C(i) = \frac{f_{c_i}}{F_i} \tag{12}$$

F_i はアプリケーション種別 i のフローの総数、そのうち、正しくアプリケーション種別 i のクラスタに分類されたフロー数を f_{c_i} とする。

また、平均正確度 C を以下のように定義する。

$$C = \frac{f_c}{F} \tag{13}$$

ここで、F はフローの総数、f_c は正しく分類されたフロー数である。

計算時間 本報告では、クラスタリングを行う計算時間についても比較を行う。本比較では、各クラスタリングアルゴリズムを C++ で実装し、Intel Core i7-4558U 2.80GHz プロセッサ、16GB RAM を搭載するコンピュータ上で NetBeans7.4[24] を用いて動作させた。

5.2 トラヒックデータの性質

クラスタリング手法の比較を行う前に、本評価に用いたトラヒックデータの特徴について述べる。図 1、図 2、図 3 に 4.2.2 節で定義した統計情報の、各アプリケーションの種類についての平均値を示す。

図 1 より、上下方向のペイロード付パケット数の比は Download、Live、Video で大きくなっており、これらのカテゴリでは少ないクライアントのリクエストに対して多量のレスポンスが返されていることが確認できる。一方、 Interactive と Upload では小さくなっており、クライアントが多くのリクエストをサーバに対して送信しているとわかる。このため、上下方向のペイロード付パケット数の比を用いることにより、クライアントからのリクエストが多い Web アプリケーションを識別することが可能であると考えらえる。

また、図 2 より、下り方向のペイロード付パケット到着レートの変動係数は Download と Live において小さくなっており、これらのカテゴリのフローでは一定のレートでレスポンスパケットが到着し続けていることを示している。それに対して、Interactive では利用者の操作に依存し、Upload では下り方向のパケットはほとんど存在しない。その結果、下り方向のペイロード付パケット到着レートは、時間変動が大きくなる。Video においても、下り方向のペイロード付パケット到着レートの変動係数は大きくなっている。これは、Video Live Streaming とは異なり、クライアント側のバッファの状況に応じて動画データが送られており、その送信レートは一定とはなっていないためであると考えられる。

図 3 より、上り方向のペイロード付パケット到着レートの変動係数も、下り方向のペイロード付パケットの到着レートの変動係数と同様に、Download と Live が小さく、他の種類のアプリケーションと大きく異なっていることがわかる。

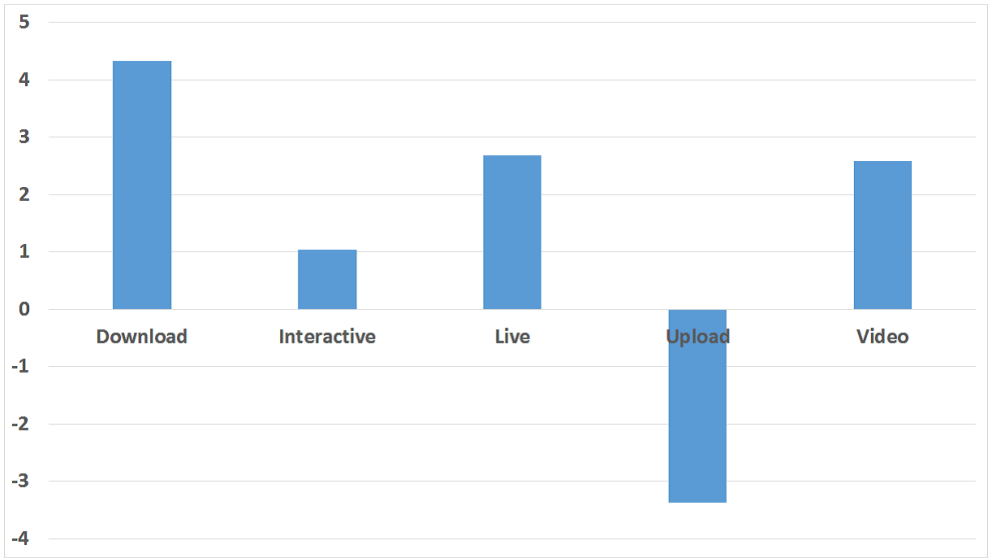

図 1: 上下方向のペイロード付パケット数の比

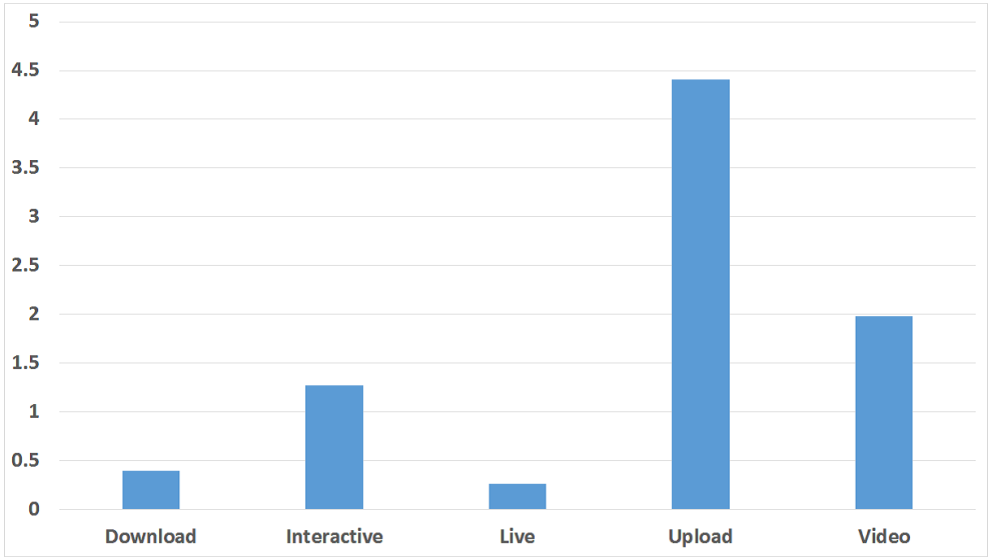

図 2: 下り方向のペイロード付パケット到着レートの変動係数

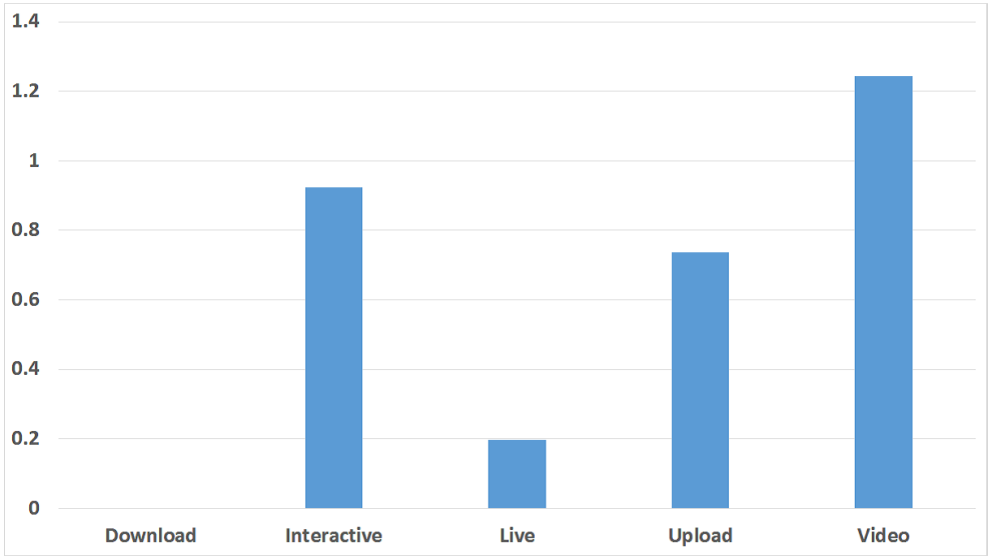

図 3: 上り方向のペイロード付パケット到着レートの変動係数

5.3 クラスタリング結果

3 章で述べた各クラスタリングアルゴリズムを用いてトラヒックのクラスタリングを行った。各アルゴリズムを用いた際のアプリケーションごとの正確度を表1、2、3、4、5、reftab:3gに示す。

この結果より、いずれのクラスタリング手法を用いた場合でも Video on Demand の識別率が低いことが分かる。これは、Video on Demand フローの内、ニコニコ動画のフローは、少ないクライアントからのリクエストで多量のサーバからのレスポンスを受け取っており、Download と同様の徴候を示しているためである。Video on Demand のアプリケーションは、帯域が空いていればより多くの動画データをダウンロードしバッファリングしようとするため、ネットワークに要求する性能要件は、Download と似ている。そのため、一部の Video on Demand のアプリケーションを Download と誤認識してしまうことのネットワーク制御への影響は大きくないと考えられる。

次に、各クラスタリングにおけるパラメータの影響について考察を行う。図 4 に AntTree におけるパラメータ α_1 を変化させた場合の正確度の変化を示す。図より、α_1 が小さいと、分類器で観測されるアプリケーションの種類数が 3 の場合は高い正確度が達成されているものの、アプリケーション数が 5 の場合は正確度が悪くなることが分かる。これは、α_1 が小さいと、各データが接続しやすくなり、クラスタの構築が抑制されるためである。その結果、構築されるクラスタ数が少なくなり、アプリケーションの種類の数が少ない場合は、その少数のクラスタで全フローを正しく分類できていたものの、アプリケーションの種類の数が大きくなると、全フローを正しく分類できない。それに対して、α_1 を 0.99 と大きくすると、各データが接続せずに独自のクラスタを作りやすくなり、同種のアプリケーションのフローのデータがまとまったクラスタを構成することが難しくなる。図より、α_1 が 0.96 の場合は、構築されるクラスタがバランスよく、アプリケーション数によらず 60% 以上の正確度を達成できることが分かる。図 5、6 に、それぞれ、K-means++、群平均法の K の値を変化させた場合の正確度の変化について示す。図より、いずれの手法も、高い正確度を得ることができる K の値は分類器で観測されたフローのアプリケーションの種類数に依存している。そのため、分類器を経由するフローのアプリケーションの種類数が未知の環境に適用することは難しい。K の値を変えながらクラスタリング結果を検証することにより、適切な K を探す手法についても検討は進められているものの [25]、計算時間が長くかかる、新たなフローが到着する度に適切な K の値を探索する必要があるという問題があり、Web トラヒックの分類器における分類に適用することは困難である。

図 7、8 に、それぞれ 3 種類のアプリケーションのトラヒックが流れている場合、5 種類のアプリケーションのトラヒックが流れている場合のクラスタリング手法間の正確度の比較

を示す。本比較では、AntTree では $\alpha_1 = 0.96$ の場合、K-means++の場合は 3 種類のアプリケーションのトラヒックが流れる場合にもっとも正確な分類ができた $K = 3$、5 種類のアプリケーションのトラヒックが流れている場合にもっとも正確なクラスタリングができた $K = 7$ の場合、群平均法では 3 種類のアプリケーションのトラヒックが流れる場合にもっとも正確な分類ができた $K = 3$、5 種類のアプリケーションのトラヒックが流れている場合にもっとも正確なクラスタリングができた $K = 11$ の場合の結果を示す。本結果より、分類器を経由するフローに含まれるアプリケーション数によらず、AntTree はすべてのアプリケーションにおいて、0.6 以上の正確度を達成している。それに対して、K-means++や群平均法は $K = 3$ とすると、3 種類のアプリケーションのトラヒックが流れている場合は、全アプリケーションに対して 1.0 の正確度での識別ができているものの、5 種類のアプリケーションのトラヒックが流れている場合には Video on Demand、Video Live Streaming、Interactive を全く識別できていない。また、$K = 7$ とした K-means++や $K = 11$ とした群平均法では、3 種類のアプリケーションのトラヒックしか流れていない場合に、Interactive なアプリケーションの正確度が 0.4 以下と低くなってしまう。これは、AntTree では構成されるクラスタ数の設定は必要なく、各データが自律的に類似したデータに接続することを繰り返すことによりクラスタが構成されるためである。その結果、AntTree では、分類器を経由するフローに含まるアプリケーションの種類数が変化しても、そのアプリケーションの種類の数に応じたクラスタを構成することができている。

最後に計算時間について、図 9 に示す。図では、$\alpha_1 = 0.96$ の AntTree、$K = 7$ の場合の K-means++と群平均法の結果を示す。本結果より、AntTree、K-means++の計算時間は短いのに対して、群平均法の計算時間は長くなっている。これは、K-means++では各データはクラスタの中心とのみ比較、AntTree ではアリが移動する際の近隣のアリに該当するデータのみと比較を行うのに対して、群平均法では全データ間での類似度の比較が必要となることが原因である。

α_1	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
Download	1	1	0.7	1	1	1	1	1	0.8
Upload	0.9	0.9	0.9	0.9	0.7	0.7	0.7	0.7	0.3
Live	0	0.3	0.9	0	1	1	0.8	0.9	0.9
Video	0	0	0.8	0.6	0.3	0.6	0.6	0.6	0.3
Interactive	0.9	0	0	0.7	1	0.8	1	1	0.6
AVERAGE	0.56	0.44	0.66	0.64	0.8	0.82	0.82	0.84	0.58

表 1: 5 種類のアプリケーションの AntTree によるクラスタリング結果

α_1	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
Download	1	1	1	1	1	1	1	1	0.5
Live	1	1	1	0.9	0.9	0.9	0.9	0.8	0.7
Interactive	1	1	1	0.6	0.6	0.6	0.4	0.4	0
AVERAGE	1	1	1	0.83	0.83	0.83	0.77	0.73	0.4

表 2: 3 種類のアプリケーションの AntTree によるクラスタリング結果

K	3	5	7	9	11	13	15
Download	1	1	1	1	1	0.6	1
Upload	0	0.9	0.8	0.7	0.3	0.7	0.7
Live	0	1	1	1	0.9	0.9	0.9
Video	0	0	0.5	0.3	0.7	0.3	0.3
Interactive	0.9	1	1	1	1	1	1
AVERAGE	0.38	0.78	0.86	0.8	0.78	0.7	0.7

表 3: 5 種類のアプリケーションの K-means++によるクラスタリング結果

K	3	5	7	9	11	13	15
Download	1	1	1	1	1	1	0.4
Live	1	0.8	1	0.8	0.4	0.7	0.8
Interactive	1	0.8	0.4	0.4	0.4	0	0.4
AVERAGE	1	0.87	0.8	0.73	0.6	0.57	0.53

表 4: 3 種類のアプリケーションの K-means++によるクラスタリング結果

K	3	5	7	9	11	13	15
Download	1	1	1	1	1	0.6	1
Upload	0.9	0.9	0.8	0.7	0.3	0.7	0.7
Live	0	1	1	1	0.9	0.9	0.9
Video	0.9	0	0.5	0.3	0.7	0.3	0.3
Interactive	0	1	1	1	1	1	0.6
AVERAGE	0.38	0.78	0.86	0.8	0.7	0.7	0.7

表 5: 5 種類のアプリケーションの群平均法によるクラスタリング結果

K	3	5	7	9	11	13	15
Download	1	1	1	1	1	1	0.4
Live	1	0.8	1	0.8	0.4	0.7	0.8
Interactive	1	0.8	0.4	0.4	0.4	0	0.4
AVERAGE	1	0.87	0.8	0.73	0.6	0.57	0.53

表 6: 3 種類のアプリケーションの群平均法によるクラスタリング結果

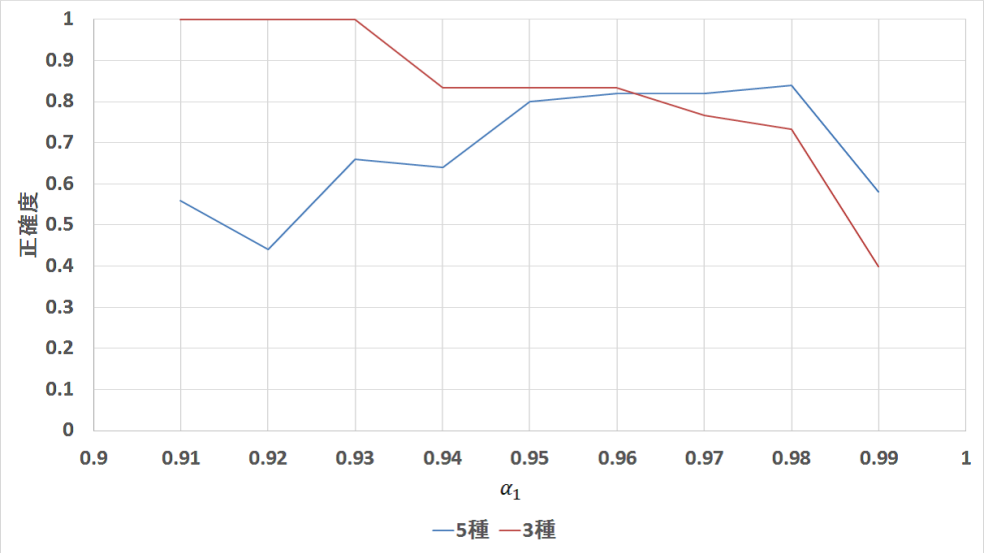

図 4: α_1 を変化させた場合の正確度の変化

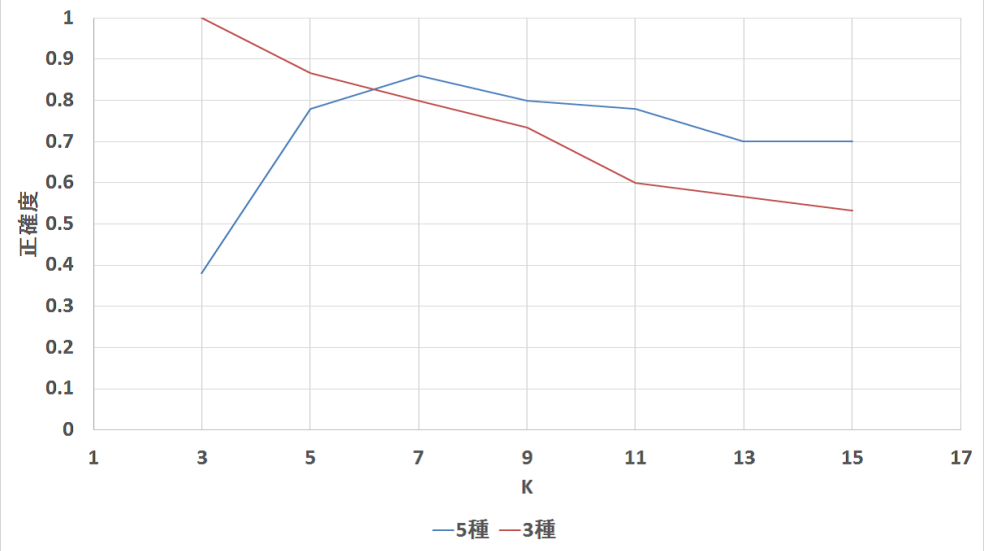

図 5: K の値を変化させた場合の正確度の変化: K-means++法

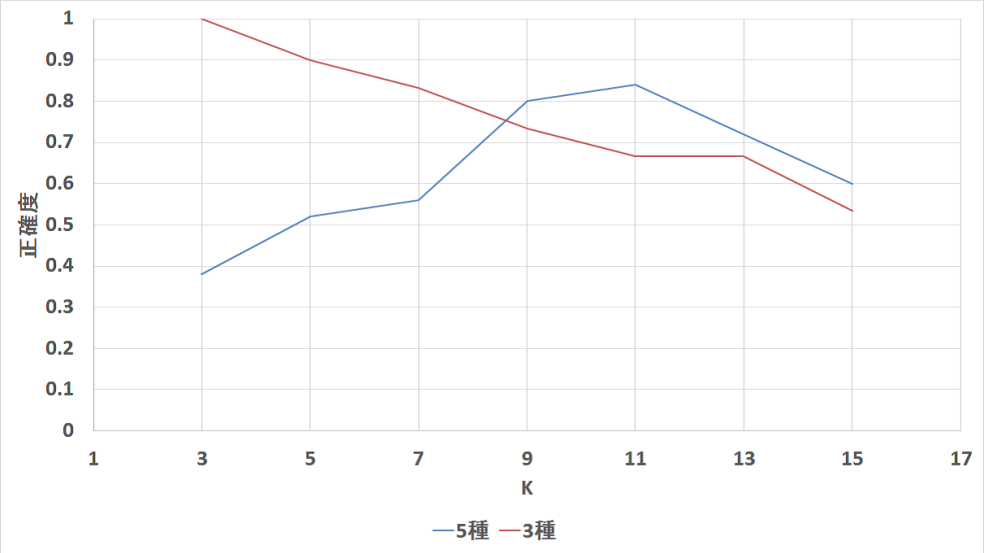

図 6: K の値を変化させた場合の正確度の変化: 群平均法

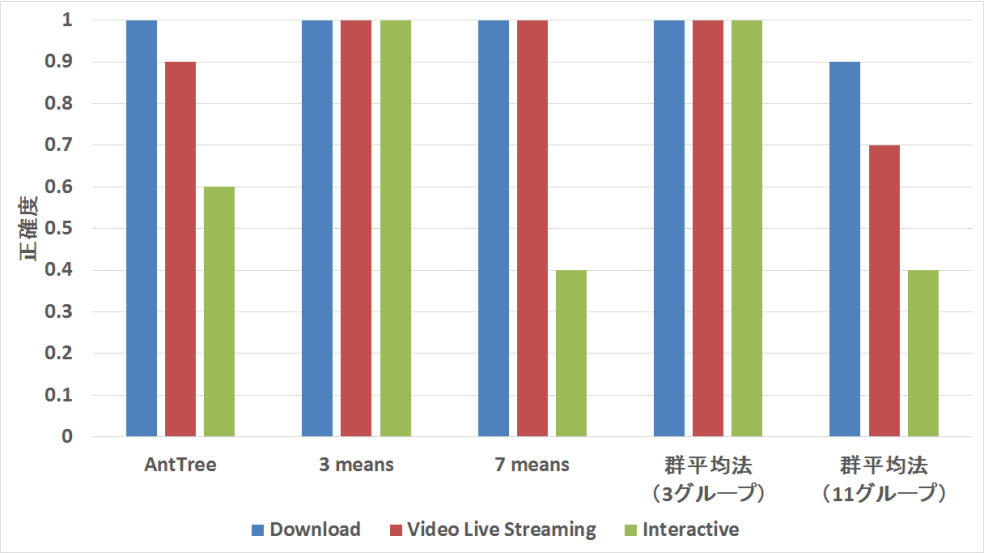

図 7: 3 種類のアプリケーションのトラヒックが流れている場合のクラスタリング手法間の正確度の比較

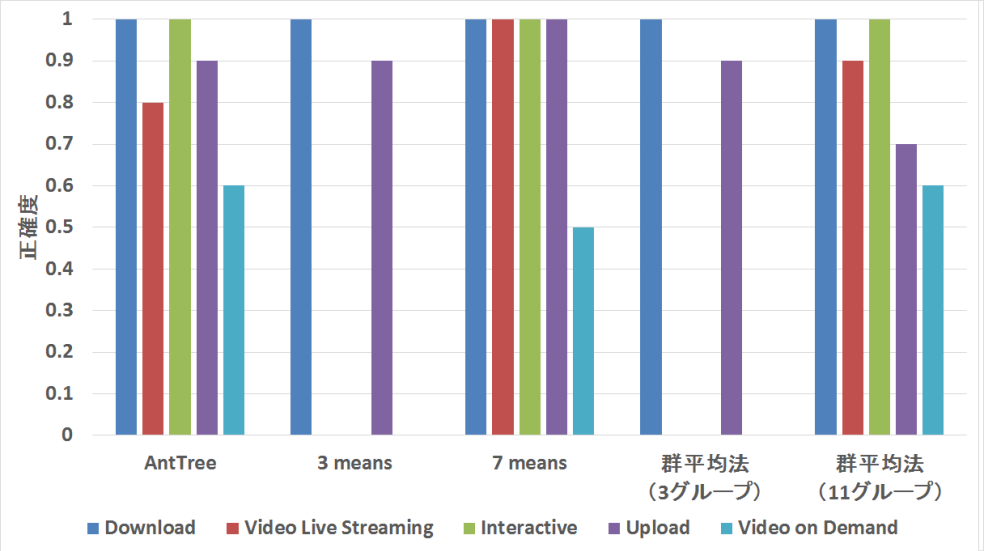

図 8: 5 種類のアプリケーションのトラヒックが流れている場合のクラスタリング手法間の正確度の比較

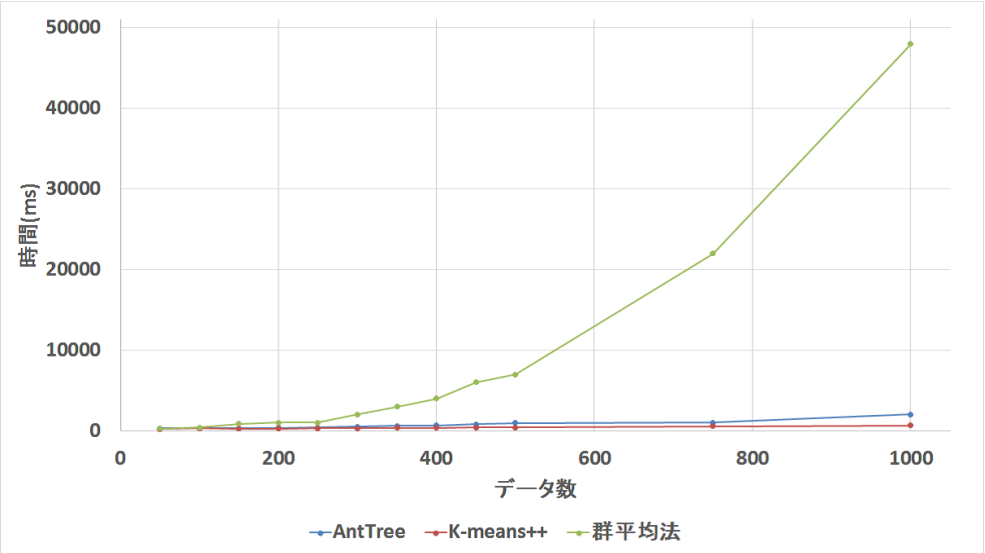

図 9: 計算時間の比較

6　おわりに

本報告では、 Web アプリケーションによるトラヒックの分類に焦点をあて、トラヒック分類器に群知能にもとづいた手法を含む複数のクラスタリング手法を適用し、それらの比較評価を行った。その結果、従来型のクラスタリング手法である K-means や群平均法は、パラメータを分類対象のデータ内に含まれているトラヒックの種類の数に合わせて設定しないと正確な分類ができないのに対して、群知能にもとづいたクラスタリング手法である AntTree は、単一のパラメータ設定により分類対象のデータ内に含まれているトラヒックの種類の数によらず、すべてのトラヒックに対して 60% 以上の割合でフローを正しいクラスタに分類できることが分かった。また、AntTree の計算時間は十分短く、1000 個のフローデータであっても、2 秒以内にクラスタリングを行うことができることが分かった。

また、AntTree では、次々とフローが到達する環境においても、アリが次々と到着し自律的に接続先のアリを選択するという形で、オンラインでクラスタリングを行うように拡張することが容易である。しかしながら、オンライン型のトラヒック分類手法への AntTree を適用した場合の評価は今後の課題である。また、オンライン型のクラスタリング手法を行うためには、分類されたクラスタのラベルを自動的に付与する手法も重要となる。今後、分類されたクラスタのラベルを自動的に付与する手法も含め、オンラインでトラヒック分類を行い、ネットワーク制御の入力として用いる手法の検討を行う予定である。

謝辞

本報告を終えるにあたり、ご指導、ご教授を賜りました大阪大学大学院情報科学研究科の村田正幸教授に厚くお礼申し上げます。また平素より丁寧なご指導を頂きました、大阪大学大学院情報科学研究科の大下裕一助教に心よりお礼申し上げます。大阪大学大学院情報科学研究科の荒川伸一准教授にも適切なご助言を頂きました。深く感謝いたします。最後に、さまざまなご助言とご助力を頂きました、大歳達也氏、辻喜宏氏をはじめとする村田研究室の方々に、お礼申し上げます。

参考文献

[1] Takashi Miyamura, Yuichi Ohsita, Shin'ichi Arakawa, Yuki Koizumi, Akeo Masuda, Kohei Shiomoto, and Masayuki Murata, "Network virtualization server for adaptive network control," in *Proceedings of 20th ITC Specialist Seminar on Network Virtualization - Concept and Performance Aspects*, May 2009.

[2] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *Communications Surveys & Tutorials, IEEE*, vol. 11, pp. 37–52, Aug. 2009.

[3] L. Popa, A. Ghodsi, and I. Stoica, "Http as the narrow waist of the future internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, (New York, NY, USA), pp. 6:1–6:6, ACM, 2010.

[4] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: A statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, (New York, NY, USA), pp. 135–148, ACM, 2004.

[5] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 50–60, June 2005.

[6] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT Conference*, CoNEXT '06, (New York, NY, USA), pp. 6:1–6:12, ACM, 2006.

[7] A. Abraham, C. Grosan, and V. Ramos, **群知能とデータマイニング**. 東京電機大学出版局, July 2012. (栗原 聡, 福井 健一：訳).

[8] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, (New York, NY, USA), pp. 512–521, ACM, 2004.

[9] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "Automated construction of application signatures," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data*, MineNet '05, (New York, NY, USA), pp. 197–202, ACM, 2005.

[10] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 23–26, Apr. 2006.

[11] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 281–286, 2006.

[12] L. Yingqiu, L. Wei, and L. Yunchun, "Network traffic classification using k-means clustering," *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on*, pp. 360–365, Aug. 2007.

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, (Philadelphia, PA, USA), pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[14] H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, "Anttree: a new model for clustering with artificial ants," *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 4, pp. 2642–2647, Dec. 2003.

[15] "Wireshark." http://www.wireshark.org.

[16] "Bitcasa." https://www.bitcasa.com.

[17] "Box." https://www.box.com.

[18] "Dropbox." https://www.dropbox.com.

[19] "Ustream." http://www.ustream.tv.

[20] "Youtube." http://www.youtube.com/.

[21] "ニコニコ動画." http://www.nicovideo.jp/.

[22] "Google マップ." https://maps.google.co.jp.

[23] “Yahoo!地図.” http://map.yahoo.co.jp.

[24] “Netbeans.” https://ja.netbeans.org/.

[25] D. Pelleg and A. W. Moore, “X-means: Extending k-means with efficient estimation of the number of clusters,” in *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML ’00, (San Francisco, CA, USA), pp. 727–734, Morgan Kaufmann Publishers Inc., 2000.