トラヒックの特徴を用いたアプリケーションの識別手法の検討

磯﨑裕臣　小川千恵 *　塩月宏次朗 †　山本神瑛 ‡　狩野悠貴 §　和田成紀 ¶

Identification method of application based on characteristics of traffic

Hiroomi ISOZAKI　Chie OGAWA*　Kojiro SHIOTSUKI†　Shinei YAMAMOTO‡
Yuki KANO§　Shigenori WADA¶

Abstract

In recent years, the lot of applications using the network are increasing. For this reason, security on the network are emphasized. Major security is to filter traffic by IP address or port number. But applications using unfiltered port numbers are increasing. So it is dificult to filter these applications. Generally, in order to identify and filter applications, it is best way to browse the contents of traffic. However, privacy protection issue remains.

Therefore in this research, we pay attention to the fact that traffic behaviors differ in the same way as the content of traffic differs depending on the application. Then we characterize the traffic behavior based on application. Finally we examine some threshholds to identify applications without browsing the contents of traffic.

Key words: Bursty flow, Pakct inter arival time, Behavior of Application

1　まえがき

近年、スマートフォンやタブレット、IoT 機器の普及により、ネットワークを利用するアプリケーションが増加している。それに伴いネットワークセキュリティの向上が重要となりつつある。セキュリティの向上として攻撃に代表される不要なトラヒックを取り除くには、一般的な IP アドレスやポート番号を用いたフィルタリングに加え、トラヒックの中身を閲覧し、トラヒックの必要性を判断するのが有効とされている。しかし、個人情報の保護やプライバシーの観点で望ましくなしい。また、暗号化通信の増加により、トラヒックの中身の閲覧自体が困難になりつつある。このため、トラヒックの中身を閲覧することなく、不要なトラヒックを識別する方法が必要とされている。

これまでに、[1] で、アプリケーションの種類やユーザ操作によってトラヒックの特徴が異なることが示されている。このため、本研究では、複数のアプリケーションのトラヒックを分析し、特徴を明らかにし、それぞれの特徴からトラヒックの中身を閲覧することなく、アプリケーションの識別を行う手法を検討する。そして、この識別手法を基にネットワーク上のトラヒックの可視化を行うシステムの構築を目標とする。

以下 2 章ではトラフィックについての説明、3 章でトラヒックの特徴を検証するためのトラヒックの測定方法、4 章にて測定結果と結果に対する考察について示す。次に 5 章で、可視化アプリケーションについて説明する。最後に 6 章でまとめと今後の課題について述べる。

2　トラヒックについて

トラヒックとはネットワークを経由する通信データのことを意味し図 1 のような構造になっている。図に示すように、

- パケット
  通信ネットワークを流れるデータの単位で、伝送されるデータ本体に送信先の所在データなど制御情報を付加した小さなまとまりのこと
- フロー
  宛先 IP アドレスとポート番号、送信元 IP アドレスとポート番号が等しいパケットの集合体である。

で構成されている。　本研究では、特徴値として、[2] に示されているフローの継続時間、フローの長さ、フローのサイズ、フロー内のパケットの最大サイズ、最小サイズ、平均サイズ、パケットの到着間隔 (Inter-Arrival Time; IAT) の最大、最小、平均の 9 要素を用いる。これらの特徴値を分析することでアプリケーションごとのトラヒックの特徴を明らかにする。

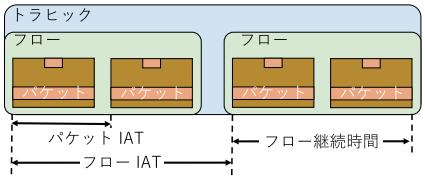


図 1: トラヒックの構造図

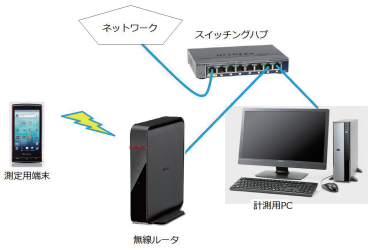


図 2: 計測環境

3　トラヒックの計測

トラヒックの計測は、スイッチングハブに無線 LAN アクセスポイントと計測用 PC を接続し、スイッチングハブの

*JFE プラントエンジ株式会社
†大阪シーリング印刷株式会社
‡株式会社エヌ・ティ・ティネオメイト
§熊本大学情報電気電子工学科
¶ハイウェイ・トール・システム株式会社

ポートミラーリング機能を用いて、無線 LAN に接続されたスマートフォンのトラヒックを計測用 PC で取得できるようにすることで実現する。計測環境のネットワーク構成を図 2 に示す。また、計測 PC でのトラヒックの計測に

4 特徴値の分析

本研究では、アプリケーションの数が豊富なゲームアプリケーションと動画アプリケーションを対象に分析を行った。対象としたアプリケーションは、

カードゲームアプリケーション

- Hanafuda・・・愛本堂花札入門 [3]
- Hearthstone・・・ハースストーン [4]

箱庭育成アプリケーション

- Garden・・・ピーターラビットガーデニング [5]

パズルゲームアプリケーション

- Pazudora・・・パズル&ドラゴン [6]
- Tsumtsum・・・ディズニーツムツム [7]

アクションゲームアプリケーション

- Shironeko・・・白猫プロジェクト [8]

ロールプレイングゲームアプリケーション

- Monst・・・モンスターストライク [9]
- Shirodora・・・城とドラゴン [10]

動画アプリケーション

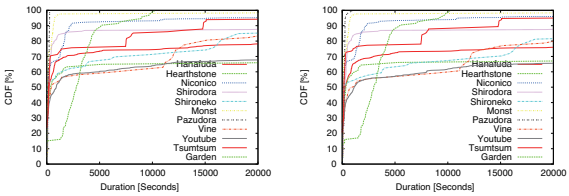
- Youtube・・・Youtube [11]
- Niconico・・・ニコニコ動画 [12]
- Vine・・・Vine [13]

の 11 種類である。これらのアプリケーションを操作した時のトラヒックを計測し、2 節で述べた 9 つの特徴値を分析する。ここでは、計測は、それぞれ 10 回ずつ行い、その平均値を特徴値として用いた。以下、4.1 から 4.9 節 では、それぞれフローの継続時間、フローの長さ、フローのサイズ、フロー内パケットの最小サイズ、フロー内パケットの最大サイズ、フロー内パケットの平均サイズ、パケット IAT の最小値、パケット IAT の平均値について検討を行う。

4.1 フローの継続時間

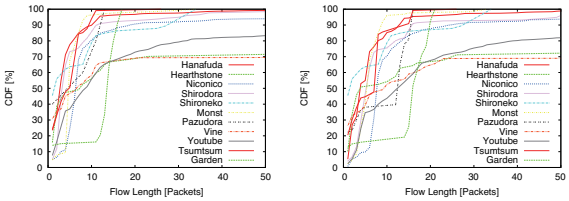
まず、フローの継続時間を考える。図 3(a) にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のフローの継続時間分布を示す。図の横軸はフローの継続時間を示し、縦軸はフロー数の累積分布 (CDF) を示している。

図 3(b) より、Garden だけが他のアプリケーションと傾向が大きくことなっていることがわかる。また、Tsumtsum も階段状の傾向を示していることより、特定の継続時間のフローで構成されていると考えられる。他のアプリケーションに関し



(a) クライアント → サーバー (b) クライアント ← サーバー

図 3: フローの継続時間



(a) クライアント → サーバー (b) クライアント ← サーバー

図 4: フローの長さ

ては、継続時間が比較的短いフローが大部分を占めており、同じような傾向を示していることがわかる。このため、特徴的である Garden と Tsumtsum は、この特徴値から識別可能であると考えられる。

4.2 フローの長さ

次にフローを構成するパケットの総数について考える。

図 4 にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のフロー内パケット数分布を示す。図の横軸はフロー内パケット数を示し、縦軸はフロー数の累積分布 (CDF) を示している。

図 4(b) より、Garden と Pazudora の傾向が他のアプリケーションと異なっていることがわかる。他のアプリケーションのトラヒックは、10 パケット未満のフローで大部分が構成されていおり、同じような傾向を示していることがわかる。

このため、特徴的である Garden と Pazudora は、この特徴値から識別可能であると考えられる。

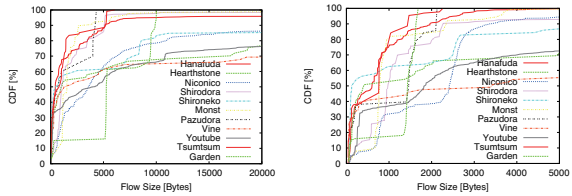
4.3 フローのサイズ

次にフローのサイズである総バイト数について考える。

図 5 にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のフローの総バイト数数分布を示す。図の横軸はフローのサイズを示し、縦軸はフロー数の累積分布 (CDF) を示している。

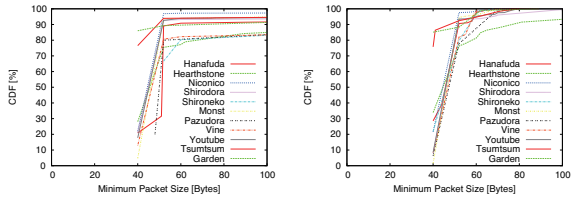
図 5(b) より、Garden のトラヒックは、1500 バイトのフローで大半が構成されていることがわかる。また、Shironeko や Niconico が階段状の傾向を示しており、特定サイズのフローで構成されていることがわかる。

このため、Garden は、1500 バイトのフローがどの程度含まれているかで識別可能であると考えられる。Shironeko と Niconico も Garden と同様に識別できる可能性があるが、2,000



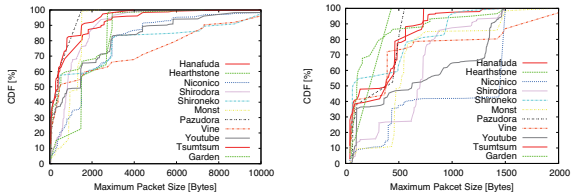
(a) クライアント → サーバー (b) クライアント ← サーバー

図 5: フローのサイズ



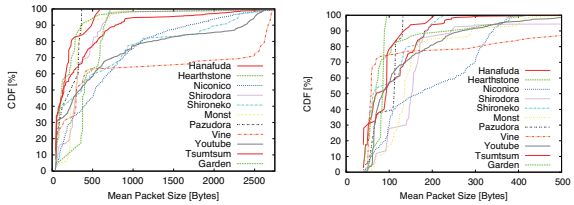
(a) クライアント → サーバー (b) クライアント ← サーバー

図 6: フロー内パケットの最小サイズ



(a) クライアント → サーバー (b) クライアント ← サーバー

図 7: フロー内パケットの最大サイズ



(a) クライアント → サーバー (b) クライアント ← サーバー

図 8: フロー内パケットの平均サイズ

バイトから 3,000 バイトの区間で同じように増加しているため、識別は難しいを考えられる。

4.4 フロー内パケットの最小サイズ

次にフロー内パケットの最小サイズについて考える。ここで、フロー内パケットの最小サイズとは、フローを構成するパケットのうちバイト数が最小のパケットのことである。

図 6 にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のフロー内パケットの最小サイズ分布を示す。図の横軸はフロー内パケットの最小サイズを示し、縦軸はフロー数の累積分布 (CDF) を示している。

図 6(b) より、Hanafuda と Hearthstine の傾向が異なることがわかるが、他のアプリケーションは、ほぼ同じような傾向を示していることがわかる。

このため、この特徴値では、Hanafuda か Hearthstine のいずれかであるか、そうでないかの識別は可能だが、特定のアプリケーションを識別することは難しいと思われる。

4.5 フロー内パケットの最大サイズ

次にフロー内パケット数の最大サイズについて考える。ここで、フロー内パケットの最大サイズとは、フローを構成するパケットのうちバイト数が最大のパケットのことである。図 10 にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のフロー内パケットの最大サイズ分布を示す。図の横軸はフロー内パケットの最大サイズを示し、縦軸はフロー数の累積分布 (CDF) を示している。

図 7(b) より、Niconico と Shirodora と Youtube の傾向がことなっていることがわかる。Garden トラヒックは、500 バイト弱のフローが多いことがわかる。また、Niconico と Vine と Youtube といった動画アプリケーションは、他のアプリケーションと異なり、大きなサイズのパケットが多く利用されていることとが分かる。さらに、Hanafuda と Hearthstone、Pazudora と Tsumtsum、Monst と Shirodora のようにゲームの種類によっても同じような傾向を示していることがわかる。

このことより、Niconico と Shirodora と Youtube と Garden は、この特徴値で識別可能であると考えられる。

4.6 フロー内パケットの平均サイズ

次にフロー内パケット数の平均サイズについて考える。ここで、フロー内パケットの平均サイズとは、フローを構成するパケットの平均バイト数のことである。

図 11 にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のフロー内パケットの平均サイズ分布を示す。図の横軸はフロー内パケットの最大サイズを示し、縦軸はフロー数の累積分布 (CDF) を示している。

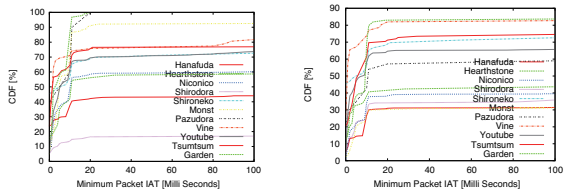
図 8(b) より、Garden と Pazudora のトラックを構成するフローは、他のアプリケーションに比べかなり小さいことがわかる。

このため、Garden と Pazudora は、トラヒックを構成するフロー内パケットの平均サイズの上限を判断することで識別可能だと考えられる。

4.7 パケット IAT の最小値

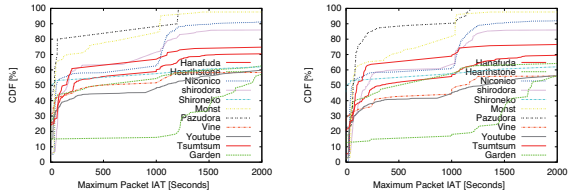
次にパケット IAT の最小値について考える。ここで、パケット IAT の最小値とは、フロー内パケットの生成間隔もしくは到着間隔の最小値のことである。

図 9(b) にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎の



(a) クライアント → サーバー (b) クライアント ← サーバー

図 9: パケット IAT の最小値



(a) クライアント → サーバー (b) クライアント ← サーバー

図 10: パケット IAT の最大値

パケット IAT の最小値分布を示す。図の横軸はパケット IAT の最小値を示し、縦軸はフロー数の累積分布 (CDF) を示している。

図 9(b) より、Vine は、他のアプリケーションに比べパケット IAT が短いフローが多いことがわかる。他のアプリケーションは、同じような パケット IAT であるフローが多いことがわかる。

Vine は、この特徴値のみで識別可能であると考えられるが、他のアプリケーションの識別は困難である。

4.8 パケット IAT の最大値

次にパケット IAT の最大値について考える。ここで、パケット IAT の最大値とは、フロー内パケットの生成間隔もしくは到着間隔の最大値のことである。

図 9(b) にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎のパケット IAT の最大値分布を示す。図の横軸はパケット IAT の最大値を示し、縦軸はフロー数の累積分布 (CDF) を示している。

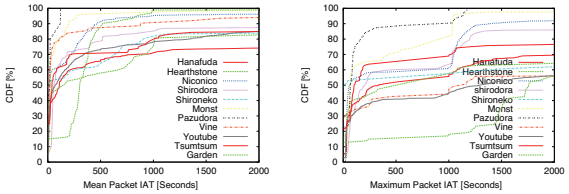
図 10(b) より、Garden と Pazudora の傾向が特徴的であることがわかる。他のアプリケーションは、同じような傾向を示している。

このため、Garden と Pazudora の識別は可能であると考えられる。

4.9 パケット IAT の平均値

最後にパケット IAT の平均値について考える。ここで、パケット IAT の平均値とは、フロー内パケットの生成間隔もしくは到着間隔の平均値のことである。

図 11(b) にクライアントからサーバへのトラヒック、サーバからクライアントへのトラヒックのアプリケーション毎の



(a) クライアント → サーバー (b) クライアント ← サーバー

図 11: パケット IAT の平均値

パケット IAT の平均値分布を示す。図の横軸はパケット IAT の平均値を示し、縦軸はフロー数の累積分布 (CDF) を示している。

図 11(b) より、Garden と Pazudora の傾向が特徴的であることがわかる。これは、前節の IAT の最大値がこの平均値も影響し、同じような傾向をしめしていると考えられる。このことより、IAT の値は、最大値に近いものが多いと考えらえる。

このため、前節と同様に Garden と Pazudora の識別は可能であると考えられるが、他のアプリケーションの識別は、困難である。

4.10 特徴値を用いた識別

前節までの検討に基づき、アプリケーションの識別方法を検討する。前節までの検討で、Garden は、ほぼ全ての特徴値についても他のアプリケーションと異なる傾向を示すことで、識別可能だと考えられる。また、Hanafuda, Shironeko, Monst 以外のアプリケーションに関してもいずれかの特徴値を用いることで識別できる可能性が高いと考えらえる。

表 1 から表 3 に特徴値の傾向の中で、特徴的なものをまとめる。表 1 は、フローについての特徴値、表 2 は、フロー内パケットについての特徴値、表 3 は、フロー内パケットの IAT についての特徴値をそれぞれまとめている。前節で述べたアプリケーション識別に利用可能と考えらえる傾向を斜体で示している。

表 1 から表 3 より 前節までの検討でいずれかの特徴値を用いることで識別可能と考えられる Hanafuda, Shironeko, Monst 以外のアプリケーションも複数の特徴値を用いることで識別精度の向上が考えらえる。また、一つの特徴値だけでは識別が困難な Hanafuda, Shironeko, Monst もより多くの特徴値を用いることで識別可能となると考えられる。一方で、Hanafuda と Hearthstone のように複数の特徴値を用いても識別が難しいことも考えらえる。しかし、これらのアプリケーションは、同じジャンルのアプリケーションであるため、トラヒックの制限といった制御を行うことについては詳細なアプリケーションの識別に限界があっても十分有効であると考えらえる。

5 トラヒック可視化システムの実現

本節では、前節までの識別方法に基づいたトラヒックの可視化するシステムを提案する。WhireShark に代表されるネットワークのトラヒックの可視化や分析が可能なアプリケーションがいくつか開発されているが、どのアプリケーションもネットワークの知識を持ったユーザ向けのものである。また、ネットワークセキュリティを実現するファイヤウォールの設定に関

表 1: 特徴値 (フローの継続時間、フローの長さ、フローのサイズ)

アプリケーション	フロー		
	継続時間	長さ	サイズ
Hanafuda	1,150 秒未満が全フローの 65% 程度		
Hearthstone	1,500 秒未満が全フローの 60% 程度	4 パケット未満のフローが全体の 50% 程度	380 バイト未満が全フローの 50 % 程度
Niconico	384 秒未満が全フローの 65% 程度、1,115 秒以上 2,307 秒未満が全フローの 25 % 程度	6 パケット以上 9 パケット未満が全フローの 65 % 程度	*2,238* バイト以上 *2,858* バイト未満が全フローの *45 %* 程度
Shirodora	796 秒未満が全フローの 85% 程度	2 パケット以上 7 パケット未満が全フローの 65 % 程度	904 バイト以上 1,142 バイト未満が全フローの 38 % 程度
Shironeko	384 秒未満が全フローの 5% 程度	1 パケットのフローが全体の 45 % 程度	*95.2* バイト未満が全フローの *38 %* 程度
Monst	769 秒未満が全フローの 96% 程度	2 パケット以上 7 パケット未満のフローが全体の 65 % 程度	619 バイト以上 1,000 バイト未満が全フローの 84 % 程度
Pazudora	384 秒未満が全フローの 96% 程度	*12* パケット以上 *16* パケット未満のフローが全体の *60 %* 程度	238 バイト未満が全フローの 38 % 程度、1,428 バイト以上 1,666 バイト未満が全フローの 84 % 程度
Vine	384 秒未満が全フローの 40% 程度、5,961 秒以上 7,307 秒未満が全フローの 13% 程度	12 パケット以上 16 パケット未満のフローが全体の 60 % 程度	381 バイト未満が全フローの 34 % 程度
Youtube	384 秒未満が全フローの 40% 程度	12 パケット以上 17 パケット未満のフローが全体の 60 % 程度	381 バイト未満が全フローの 34 % 程度
Tsumtsum	*384* 秒未満が全フローの *73%* 程度、*7,692* 秒以上 *8,461* 秒未満が全フローの *10%* 程度、*14,807* 秒以上 *15,192* 未満のフローが全体の *5 %* 程度		190 バイト未満が全フローの 38 % 程度、1,428 バイト以上 1,666 バイト未満が全フローの 38 % 程度
Garden	*1,500* 秒以上 *5,000* 秒未満が全フローの *75 %* 程度	*13* パケット以上 *21* パケット未満のフローが全体の *85 %* 程度	*1,333* バイト以上 *1,666* バイト未満が全フローの *83 %* 程度

表 2: 特徴値 (フロー内パケットの最小・最大・平均サイズ)

アプリケーション	フロー内パケット		
	最小サイズ	最大サイズ	平均サイズ
Hanafuda	*40* バイトが全フローの *75 %* 程度	76 バイト未満が全フローの 38 % 程度	66 バイト未満が全フローの 50 % 程度
Hearthstone	*40* バイトが全フローの *85 %* 程	142 バイト程度が全フローの 45 % 程度	76 バイト未満が全フローの 70 %程度
Niconico		*1,500* バイトが全フローの *60 %* 程度	
Shirodora		*615* バイト以上 *769* バイト未満が全フローの *48 %* 程度	138 バイト以上 176 バイト未満が全フローの 55 % 程度
Shironeko		39 バイト未満が全フローの 50 % 程度	52 バイト以上 62 バイト未満が全フローの 50 % 程度
Monst		461 バイト以上 615 バイト未満が全フローの 68 % 程度	66 バイト以上が全フローの 80 % 程度
Paudora		97 バイト未満が全フローの 38 % 程度	119 バイト程度が全フローの 45 % 程度
Vine		77 バイト未満が全フローの 38 % 程度	
Youtube		*1,307* バイト以上 *1500* バイト未満が全フローの *33 %* 程度	
Tsumtsum	41 バイト未満が全フローの 89 % 程度	77 バイト未満のフローが全体の 38 % 程度	95 バイト程度が全フローの 45 % 程度
Garden	40 バイト未満が全フローの 88 % 程度	*57* バイト以上 *250* バイト未満が全フローの *85 %* 程度	*85* バイト以上 *91* バイト未満が全フローの *88 %* 程度

しても IP アドレスやポート番号を始め多くのことを細かく設定する必要がある。このため、これらのアプリケーションを利用したり設定を行ったりするには、専門的な必要となりそのような知識を持っていないユーザにとってしきいが高い。

そこで、本研究では、ネットワークに関する高度な知識を必要することなく、ネットワーク上のトラヒックの状態確認やファイヤウォールの設定を可能とするアプリケーションを提案する。具体的には、ブログ等で利用されているタグクラウドと呼ばれる投稿数に応じてカテゴリやタグの文字サイズが変化するもののように、トラヒック内のアプリケーションのパケット数や占有率に応じて文字サイズを変更したり、図形を大きくしたりする。また、特定アプリケーションの帯域制限やフィルタリングについても、該当するアプリケーションの文字や図形をゴミ箱にドラッグアンドドロップしたり、文字や図形を変形させたりすることで実現する。

実際に Windows アプリケーションを開発し、必要な機能のうち、基本的な部分の実装を行い、トラヒックを計測し、IP, TCP, UDP ヘッダを正しく取得できることを確認した。

6 おわりに

本研究では、トライヒックの中身を閲覧することなく、トラヒックの識別を実現するために、複数のアプリケーションを対象にトラヒックの計測を行い、アプリケーションごとのトラヒックの特徴を分析し、その違いを明らかにした。そして、単一の特徴値だけでは、アプリケーションの識別は不可能であるが、複数の特徴値を組み合わせることで、アプリケーションの識別が可能となることを示した。

今後の課題は、フローの長さが極端に短いものの特徴値を取得する方法の検討、トラヒックの特徴は、ネットワーク環境や端末に依存する可能性があるため、計測対象を増やし、信頼性の高い特徴値を得ることである。また、今回明らかになった特徴値を用いて実際に識別を行い、識別精度を向上させる必要がある。最終的には、可視化アプリケーションに識別手法を実装し、ネットワークの高度な知識を必要とすることなく容易にネットワークの状態やセキュリティの管理を実現可能とする必要がある。

参考文献

[1] H. Isozaki, S. Ata, and I. Oka, “Modeling of flows based on behavior of applications,” in *Proceedings of 26th International Conference on Information Networking*, pp. 447–552, February 2012.

[2] A. Moore, M. Crogan, A. W. Moore, Q. Mary, D. Zuev, D. Zuev, and M. L. Crogan, “Discriminators for use in flow-based classification,” tech. rep., 2005.

[3] Hayakawa, L.LC, “愛本堂花札 lite version 2.1.0.” http://hanafuda.richmind.net/, February 2014.

[4] Blizzard Entertainment,Inc., “Hearthstone.” `https://play.google.com/store/apps/details?id=com.blizzard.wtcg.hearthstone`, October 2015.

[5] I. Poppin Games, “ピーターラビットガーデン version 5.3.” `https://play.google.com/store/apps/details?id=com.poppingames.android.peter`, December 2015.

[6] GungHo Online Entertainment, Inc., “パズル&ドラゴンズ version 9.0.0.” `http://pad.gungho.jp/member/`, January 2016.

表 3: 特徴値 (IAT の最小・最大・平均値)

アプリケーション	パケット IAT		
	最小値	最大値	平均値
Hanafuda			
Hearthstone		5.8 秒未満が全フローの 40 %	40 秒未満が全フローの 35 %
Niconico		8 秒未満が全フローの 53 %	40 秒未満が全フローの 60 %
Shirodora		7.7 秒未満が全フローの 28 % 程度	59 秒未満が全フローの 60 % 程度
Shironeko	0 ミリ秒が全フローの 45 % 程度	0 秒が全フローの 50 % 程度	0 秒が全フローの 50 % 程度
Monst		96.2 秒未満が全フローの 60 % 程度	98 秒未満が全フローの 75 % 程度
Pazudora		*96.2* 秒未満が全フローの *70* % 程度	*20* 秒未満が全フローの *73* % 程度
Vine	*1.9* ミリ秒未満が全フローの *75* % 程度	0 秒が全フローの 28 %程度	
Youtube		0 秒が全フローの 20 %程度	
Tsumtsum		0 秒が全フローの 20 %程度、115 秒未満が全フローの 50 %	59 秒未満が全フローの 60 % 程度
Garden	8.7 ミリ秒以上 11.7 ミリ秒未満が全フローの 40 % 程度	*1,577*秒未満が全フローの *40* %程度	*235* 秒以上 *373* 秒未満が全フローの *78* % 程度

[7] LINE Corporation, “Line：ディズニー ツムツム version 1.28.0.” `https://play.google.com/store/apps/details?id=com.linecorp.LGTMTM`, January 2016.

[8] COLOPL, Inc., “白猫プロジェクト version 1.0.48.” `https://play.google.com/store/apps/details?id=jp.colopl.wcat`, January 2016.

[9] COLOPL, Inc., “モンスターストライク version 5.5.4.” `https://play.google.com/store/apps/details?id=jp.co.mixi.monsterstrike`, January 2016.

[10] mixi, Inc, “城とドラゴン version 2.1.4.” `https://play.google.com/store/apps/details?id=jp.co.asobism_castleanddragon`, January 2016.

[11] Google, Inc., “Youtube version 10.54.” `https://play.google.com/store/apps/details?id=com.google.android.youtube`, January 2016.

[12] DWANGO Co., Ltd., “niconico version 6.39.” `https://play.google.com/store/apps/details?id=jp.nicovideo.android`, October 2015.

[13] Vine Labs, Inc., “Vine version 5.7.6.” `https://play.google.com/store/apps/details?id=co.vine.android`, Febrary 2016.

(2016 年 11 月 7 日　受理)