Vol.2015-DPS-165 No.15
2015/12/11

Androidアプリケーションからの通信特徴抽出手法

中野 雄介[1,2,a)]　上山 憲昭[1,2]　塩本 公平[1]　長谷川 剛[3]　村田 正幸[2]

概要：近年，スマートフォンの普及により，モバイルネットワークに対する負荷が増大している．スマートフォンのユーザは様々なアプリケーションを自由にインストールすることができる．また，このようなアプリケーションは通信キャリアでなくても，誰もが作成・公開できる．このため，モバイルネットワークへの負荷を考慮しないアプリケーションが広く普及し，モバイルネットワークへの負荷が増大していると考えれられる．本稿では，様々なアプリケーションが混在するパケットキャプチャデータから，アプリケーションごとのパケットに分離するためのアプリケーションごとの通信の特徴を，Android アプリケーションをリバースエンジニアリングすることで生成する手法について提案する．抽出される特徴は，アプリケーションの各通信の内容とそれらの順序で構成される．提案手法の有効性を示すため，提案手法を用い，実際に公開されているインスタントメッセージングアプリケーションから通信の特徴を抽出し，そのアプリケーションが実際に送出するパケットと一致していることを確認した．

Communication Feature Extraction from Android Applications

Nakano Yuusuke[1,2,a)]　Kamiyama Noriaki[1,2]　Shiomoto Kohei[1]　Hasegawa Go[3]
Murata Masayuki[2]

Abstract: Due to expansion of smartphone, mobile network load is increasing these days. Smartphone users are allowed to install various applications into their smartphones freely. These applications are made and provided by anyone even if he/she is not a telecommunications carrier. Thus, mobile network load is thought to be increasing because of expansion of applications designed in mind the load reduction. In this paper, we propose a method for extracting communication features from Android applications by reverse engineering the applications to make it possible to classify packets sent by various applications according to applications. The extracted features have content of each packet and sequence of the packets. To evaluate the effectiveness of the proposed method, we determined if the features extracted from an Android application using the proposed method correspond to actual packets sent by the application.

1. はじめに

近年，スマートフォンの普及により，モバイルネットワークに対する負荷が増大している．スマートフォンのユーザは様々なアプリケーションを自由にインストールできる．また，スマートフォンアプリケーションは通信キャリアでなくても，誰もが作成・公開できるため，モバイルネットワークへの負荷を考慮しないアプリケーションが公開されるおそれがある．たとえば，特定時刻に通信するようなアプリケーションが普及すると，1つ1つの通信でやり取りされるパケットサイズが小さいとしても，短時間に一斉に通信開始するスマートフォンに対するネットワーク内での処理負荷が増大することが考えられる．

通信キャリアとして，モバイルネットワークに対する影響の大きいアプリケーションを早期に発見することが重要である．影響の大きさは2つの観点で決定される．1つはアプリケーションの利用者数．もう1つはアプリケーションのネットワークに対する動作である．インストール数はアプリケーションが公開されているサイト (GooglePlay

[1] 日本電信電話株式会社 NTT ネットワーク基盤技術研究所
NTT Network Technology Laboratories, NTT Corporation
[2] 大阪大学大学院情報科学研究科
Department of Information Science, Osaka University
[3] 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University
[a)] nakano.yuusuke@lab.ntt.co.jp

等) に記載されているインストール数と，モバイルネットワークの契約端末数とで大まかな数を特定できる．しかし，GooglePlay には大まかなインストール数のみが公開されているため，最大で 2 倍程度の誤差がでてしまう．また，このようなサイトで公開されていないアプリケーションについて，インストール数を知ることはできない．

アプリケーションのネットワークに対する動作とは，アプリケーションがパケットを送受信するタイミングやそのサイズである．動作の特定にはアプリケーションのソースコードの静的解析が有効である．リバースエンジニアリングツールを利用することで，Android アプリケーションのソースコードを抽出し，静的解析することでアプリケーションの動作を特定することができる．しかし，膨大なスマートフォンアプリケーションが公開されており，それらを静的解析するには膨大な稼働が必要となる．

そこで，本稿では Android アプリケーションからの通信特徴抽出手法を提案する．本手法は Android アプリケーションの配布形式である APK ファイルを解析することで，そのアプリケーションの通信の特徴を抽出する．抽出される特徴は，アプリケーションの各通信の内容とそれらの順序で構成される．

抽出された特徴を用いることで，様々なアプリケーションのパケットが混在するパケットキャプチャデータから，アプリケーションごとのパケットに分類可能となる．特に複数のアプリケーションが共通のサーバと通信している場合，1 つ 1 つの通信内容が似通うため，分類は困難であったが，提案手法は通信の順序も抽出するため，アプリケーション間で通信の順序が異なる場合は，提案手法で抽出される特徴で分類できると考えられる．このようにして，モバイルネットワーク内でのキャプチャデータをアプリケーションごとに分類することで，アプリケーションごとの利用者数を特定できる．また，利用者の増加率の高いアプリケーションに対して，分類されたアプリケーションごとのパケットの送信パターンを解析することで，たとえば，特定時刻に通信するようなアプリケーションを発見できる．

2. 関連研究

複数のアプリケーションのパケットが混在するパケットキャプチャデータから，アプリケーションごとのパケットに分類する様々な手法が提案されている．

伝統的には 5-tuple を用いた分類が利用されてきた．キャプチャデータから，送信元 IP アドレス，ポート番号，送信先 IP アドレス，ポート番号，プロトコル番号によってパケットを分類する方法である．しかし，アプリケーションの送信先 IP アドレス，ポート番号を特定する手法を新たに検討する必要がある．また，複数のアプリケーションが同一のサーバとパケットを送受信する場合もある．例えば，Twitter の API で実装されたアプリケーションは，異なるアプリケーションであっても共通の Twitter のサーバとパケットの送受信を行うと考えられる．5-tuple による分類ではこのようなパケットをアプリケーションごとに分類することは困難である．

また，5-tuple 以外の情報を用いた多くの分類手法が提案されている．しかし，このような手法は P2P などの特定のアプリケーションのトラフィックを分類する手法であり，膨大な種類のスマートフォンアプリケーションごとのパケットに分類するようなことは困難である [1], [2], [3], [4], [5], [6].

Network Log はスマートフォン内でのパケットキャプチャを実現するツールであり，キャプチャ結果をアプリケーションごとに分類することができるため，アプリケーションごとの通信の振る舞いを解析できる [7]．また，スマートフォンアプリケーション自体を解析することで，悪意のある通信を行うアプリケーションの特定が実現されている．Gibler らは Android アプリケーションをリバース・エンジニアリングすることで，ユーザ情報などを外部に送信するようなアプリケーションを特定する AndroidLeaks を提案している [8]．また，Enck らは Android アプリケーションの動作をリアルタイムにトラッキングすることで，ユーザ情報の送信などの振る舞いを発見する TaintDroid を提案している [9]．このような手法はアプリケーションの静的/動的解析により，ネットワークに対して特定のデータを送信するアプリケーションを特定することはできるため，ネットワークに対するインパクトの大きいアプリケーションの特定に応用できるかもしれない．しかし，アプリケーションの普及に関する情報は取得できない．

スマートフォンアプリケーション自体の解析結果を用い，キャプチャデータ内で特定のアプリケーションの存在を特定するためのアプリケーションごとのフィンガープリントを生成するための NetworkProfiler と呼ばれる手法が Dai らによって提案されている [10]．この手法は Android アプリケーションごとの HTTP の特徴を，アプリケーションに含まれる広告などの記述から抽出することでフィンガープリントを生成している．しかし，キャプチャデータ内で特定アプリケーションの存在を特定するためのフィンガープリントであり，アプリケーションごとにキャプチャデータを分類することはできない．

3. 提案手法

提案手法の概要を図 1 に示す．まず Soot [11] を用い，Android アプリケーションの配布形式である APK ファイルからソースコードを生成する．その後，ソースコードを解析し，ソースコードに含まれるメソッド実行時の前後関係を抽出する．この前後関係をたどりながら，通信関連メソッドを探索することで，通信関連メソッドの呼び出し順序を抽出する．最後に通信関連メソッドの引数に設定されている文字列等の値を抽出することで，各通信の内容を抽

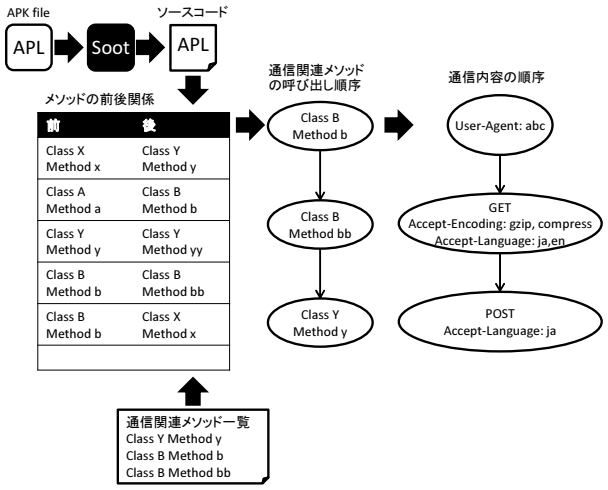

図 1　提案手法の概要
Fig. 1　Abstract of proposed method

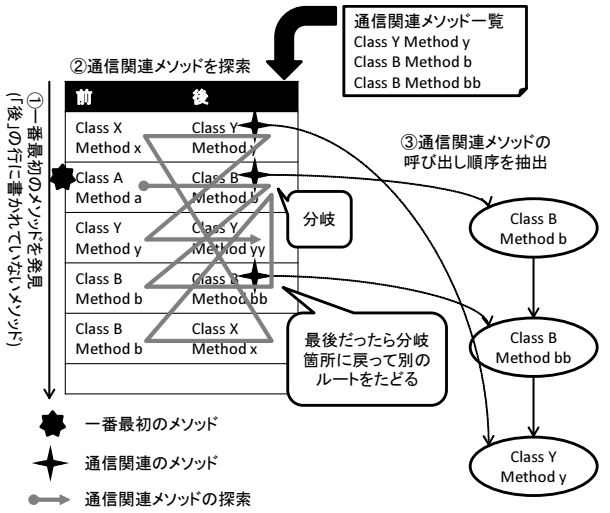

図 2　通信関連メソッドの呼び出し順序の抽出
Fig. 2　Extraction of communication methods flow

出し，通信内容の順序を生成する．

以下で，ソースコードの生成以降の各手順について詳細に説明する．

3.1　メソッドの前後関係

メソッドの前後関係を抽出するために，以下の 4 つをソースコードから発見する．

3.1.1　メソッドの呼び出し

あるメソッドが別のメソッドを呼び出している場合，あるメソッドの後で別のメソッドが実行されるという前後関係がある．このため，あるメソッド→別のメソッド という前後関係を抽出する．

3.1.2　Intent を使った呼び出し

Android アプリケーションでは，Intent と呼ばれる枠組みを用いて，画面遷移などを行う．ある画面と次の画面という前後関係を抽出するため，ソースコード内の Intent を利用している箇所を発見し，そこに設定されている画面 (Activity を継承した) クラスを抽出する．これにより，Intent が利用されているメソッド→抽出されたクラスの onCreate メソッド (Activity 生成時に最初に呼ばれるメソッド) という前後関係を抽出する．

3.1.3　Activity や AsyncTask などのライフサイクル

Android では Activity と呼ばれる画面を実装するためのクラスや，AsyncTask と呼ばれる非同期処理を実装するためのクラスが提供されている．このようなクラスを継承することで，独自の画面や非同期処理を実装することができる．このようなクラスのメソッドは予め決められたタイミング・順序で実行される．このため，予め決められた順序に沿って前後関係を抽出する．

3.1.4　イベントリスナによる実行

Android アプリケーションは Java で実装される．Java はイベントリスナと呼ばれるクラスを提供しており，そのクラスを継承して実装されたクラスは，画面のクリックなど，特定のイベントが発生した際に実行される．イベントとイベントリスナとのひも付けをしているメソッドと，イベント発生時に実行されるメソッドとの間には前後関係があるため，setOnXXXListener → OnXXXListenr.onXXX という前後関係を抽出する．なお，XXX には Click などのイベントが入り，抽出対象のイベントは予め設定ファイルで決めておく．

3.2　通信関連メソッドの呼び出し順序

通信関連メソッドの呼び出し順序の抽出手法を図 2 に示す．まず，一番最初に実行されるメソッドを発見する．このメソッドは，メソッドの前後関係内で，どのメソッドの「後」にもなっていないメソッドである．その後，予め作成されている通信関連メソッド一覧を参照しつつ，一番最初に実行されるメソッドから，後に実行されるメソッドの方に向かってメソッドの前後関係を探索することで，通信関連のメソッドを発見する．このようにして，通信関連のメソッドが発見された順番を通信関連のメソッドの呼び出し順序として抽出する．

なお，あるメソッドの後に複数のメソッドが呼ばれることで，分岐が発生している場合がある．この時，あるメソッドのソースコード内の，複数メソッドの出現順によって，複数メソッド間の前後関係を判定し，出現順の早いメソッドを先に探索する．前のメソッドから後のメソッドを辿れなくなるまで探索を進め，辿れなくなった段階で直近の分岐に戻り，複数メソッドのうち，後と判定されたメソッドの方の探索を進め，探索できるメソッドがなくなるまで探索を続ける．

3.3　通信内容の順序

最後に各メソッドの通信内容を抽出し，アプリケーショ

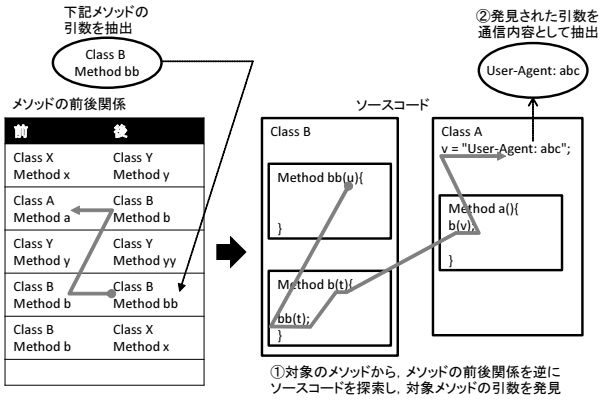

図 3　通信内容の抽出

Fig. 3　Extraction of communication characteristic

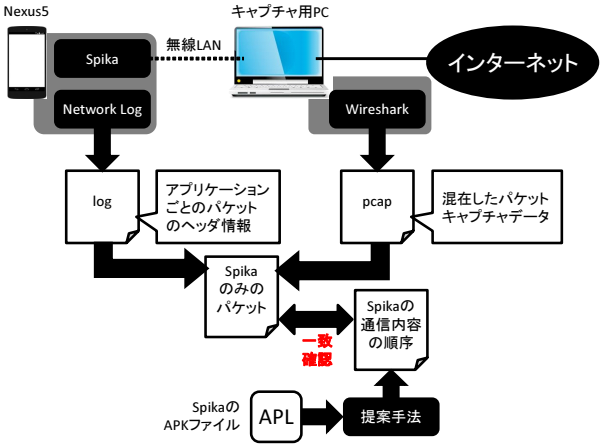

図 4　評価手法

Fig. 4　Evaluation environment

ンの通信内容の順序を生成する．提案手法は通信内容として，通信関連メソッドの引数に設定されている文字列などの値を，ソースコードから抽出する．図 3 にその手順を示す．まず，通信関連メソッドから，メソッドの前後関係の逆方向にソースコードを探索し，通信関連メソッドの引数に設定されている値が定義されている箇所を発見する．その後，発見された値を通信内容として抽出する．通信関連メソッド全てに対して，引数に設定されている値を抽出することで，通信内容の順序を生成する．

4. 評価

提案手法の有効性を評価するため，実際に公開されている Android アプリケーションから通信内容の順序を抽出し，抽出結果がそのアプリケーションのキャプチャデータの内容と一致していることを確認した．

本手法は，ソースコードを解析するためにリバースエンジニアリングを行う．このため，今回はオープンソースのインスタントメッセージングアプリケーションである Spika で提案手法の有効性を評価した．Spika はユーザ同士のテキストチャットやファイルの受け渡し，音声・映像によるコミュニケーションを提供するアプリケーションである [12]．

4.1 評価手法

図 4 に評価手法の概要を示す．提案手法で抽出された通信内容の順序が，実際のキャプチャデータの内容と一致していることを確認するため，スマートフォンで Spika を動作させた時のキャプチャデータから，Spika のみのパケットを抽出した．Spika のみのパケットの抽出結果と，提案手法による通信内容の順序が一致していれば，提案手法で抽出された特徴で，混在するパケットキャプチャデータから Spika のパケットを分類できると考えられる．

評価環境を説明する．Nexus5 に対象アプリである Spika と，スマートフォン内でアプリケーションごとのパケットをキャプチャできるアプリケーションである Network Log をインストールしておく．また，インターネットに接続されたキャプチャ用 PC で Wireshark を動作させておき，Nexus5 とキャプチャ用 PC を無線 LAN で接続する．これにより，Nexus5 はキャプチャ用 PC を経由してインターネットに接続でき，キャプチャ用 PC において Nexus5 のパケットをキャプチャできる．

Spika のキャプチャデータを収集するため，Spika 上で以下の操作を行い，スマートフォン内とキャプチャ用 PC においてパケットキャプチャした．

- 起動，放置，終了
- メッセージ送信

以上のようにしてキャプチャした結果は，スマートフォン内で生成される Network Log のログデータと，キャプチャ用 PC 内で生成される Pcap データである．ログデータはアプリケーションごとの分類されたパケットの送信先/送信元 IP アドレス，ポート番号，サイズ等のヘッダ情報である．これには，パケットのペイロードの内容は含まれていない．一方，Pcap データはパケットのペイロードの内容も含まれているが，アプリケーションごとに分類されていない．このため，これら 2 つのデータから，Spika のみのペイロードの内容も含めたパケットキャプチャデータを抽出した．

一方，提案手法を用い，Spika の APK ファイルから通信内容の順序を抽出した．この際，通信関連メソッドとして，HTTP やソケット通信に関するメソッドを抽出対象とした．この抽出結果と先で抽出された Spika のみのパケットとの一致を確認することで，提案手法が正しく Spika の通信内容の順序を抽出できていることを確認した．

4.2 評価結果

Spika の通信内容の順序を抽出した結果を図 5 に示す．Android アプリケーションの Service と呼ばれるバックグ

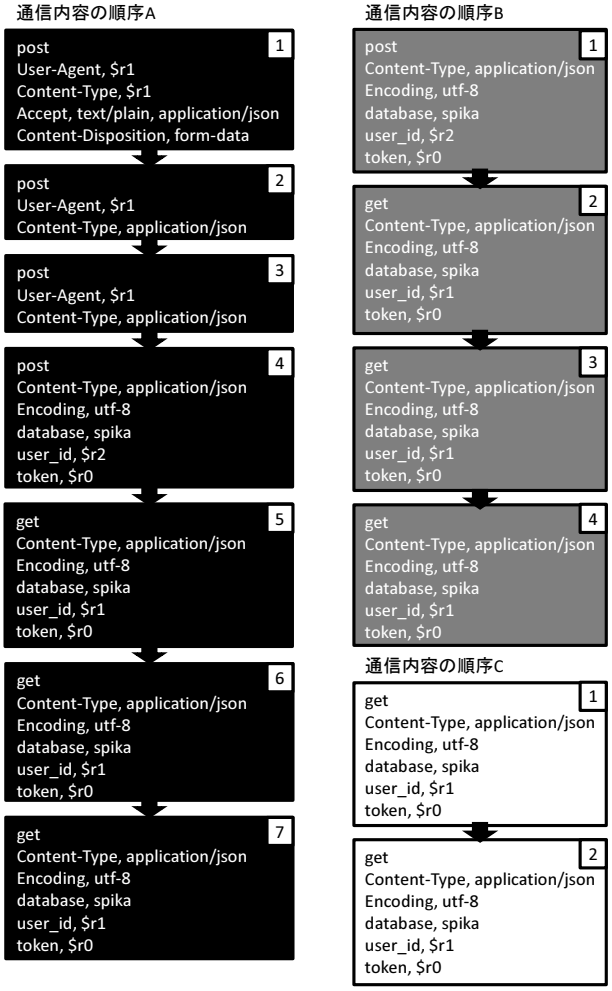

図 5　通信内容の順序の抽出結果
Fig. 5　Extraction result of sequences of packets

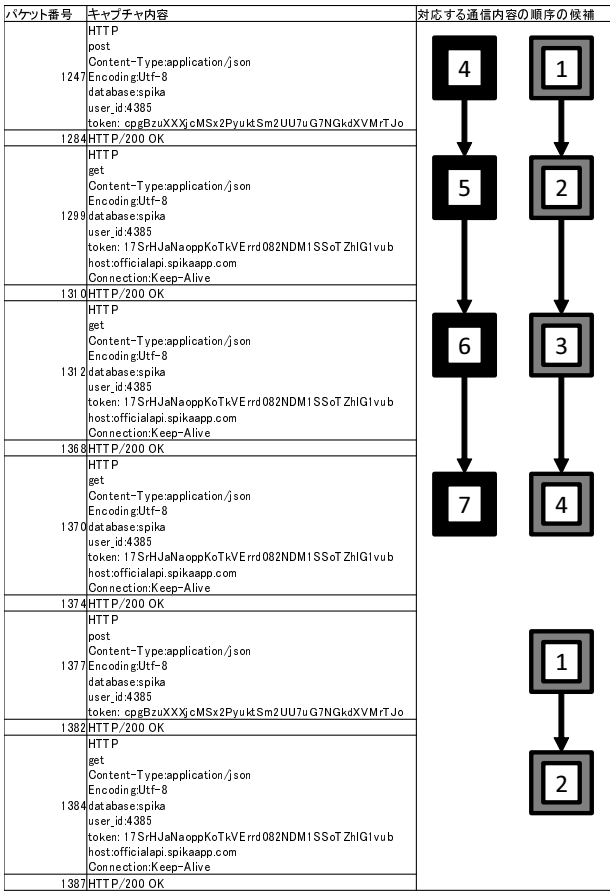

図 6　Spika を起動，放置，終了した際のパケットキャプチャ結果と通信内容の順序との対応関係
Fig. 6　Packets of Spika with launching and quitting, and extracted sequences of packets

ラウンド処理の仕組みや，今回対象外としたイベント処理などがあり，複数の順序が抽出された．各長方形はそれぞれの通信を表し，含まれる文字列は通信内容として抽出された文字列を示している．文字列中の$r0 などは変数を表し，具体的な文字列が抽出できなかった箇所である．また，説明のために送信順に番号を付与している．今回は HTTP での通信に関するメソッドとソケット通信に関するメソッドを通信関連メソッドとして抽出対象とした．

図 6 に，Spika を起動，放置，終了した際のパケットキャプチャ結果と，通信内容の順序との対応関係を示す．「キャプチャ内容」の列にはキャプチャされたパケットに含まれる HTTP のヘッダ情報を記載している．また，「対応する通信内容の順序の候補」の列には，図 5 に示す A〜B の通信内容の順序のうち，キャプチャ結果と類似する順序の，類似する箇所を番号で示している．キャプチャデータの前 4 つの HTTP については順序 A の一部，または，順序 B と一致することが分かる．また，後 2 つの HTTP については順序 B の一部と一致することが分かる．

一方，図 7 に，Spika でメッセージを送信した際のパケットキャプチャ結果と，通信内容の順序との対応関係を示す．順序 B と C とがキャプチャデータと一致していることが分かる．

以上のように，提案手法を用いて Spika から抽出した通信内容の順序と，Spika を操作することで発生するパケットとが一致することがわかった．

5. おわりに

本稿では，Android アプリケーションの配布形式である APK ファイルを解析することで，そのアプリケーションの通信の特徴を抽出する手法を提案した．提案手法を用いてオープンソースのインスタントメッセージングアプリケーションである Spika から通信の特徴を抽出した結果と，実際のパケットキャプチャ結果とから，提案手法は Spika の通信の特徴を抽出できていることがわかった．

今後は，Spika 以外の多様な Android アプリケーションから通信の特徴を抽出し，有効性を確認する．Spika は HTTP で通信を行うアプリケーションであったため，特有のヘッダ情報を含む特徴を抽出することができた．一方，一般的な Socket 通信を行うようなアプリケーションでは，

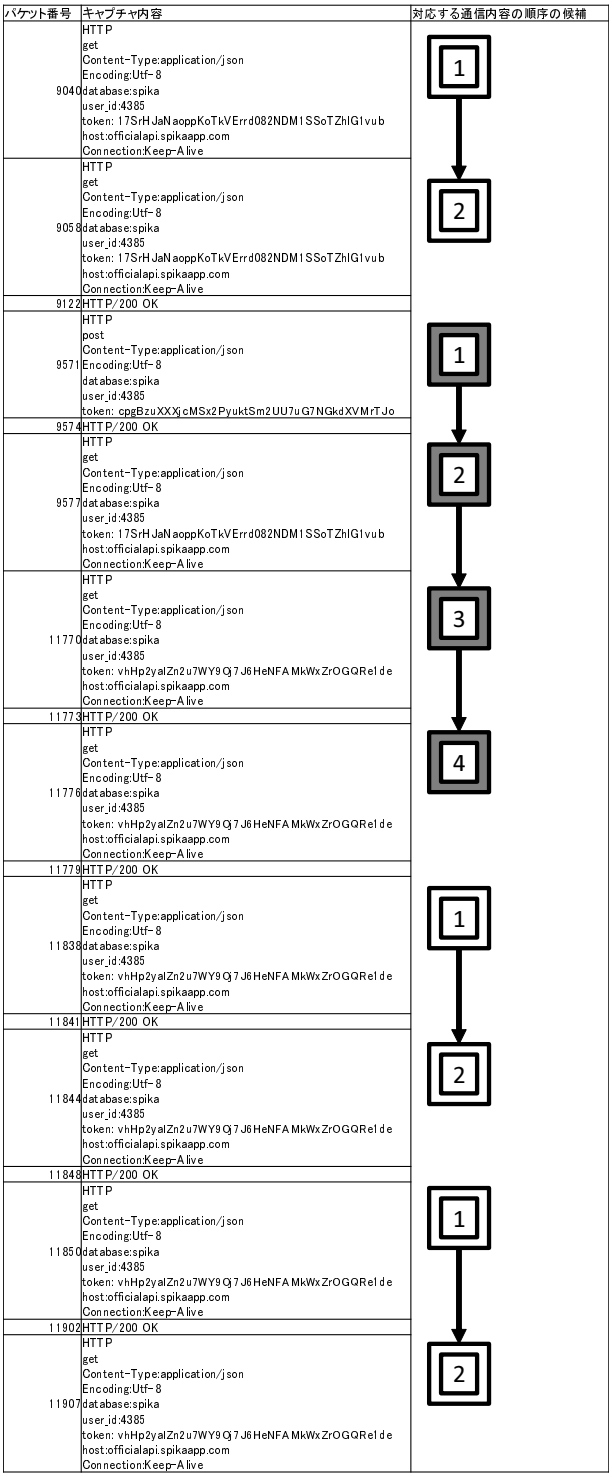

図 7 Spika でメッセージを送信した際のパケットキャプチャ結果と通信内容の順序との対応関係

Fig. 7 Packets of Spika with sending messages, and extracted sequences of packets

アプリケーションごとに特有の文字列をソースコードから抽出できない可能性がある．

また，抽出された通信の特徴を用い，複数の Android アプリケーションのパケットが混在するキャプチャデータからの，アプリケーションごとのパケットの分類手法についても検討をすすめる．今回，分岐部でのメソッドの前後関係は，ソースコード内での出現順位を用いて決めている．これは，if 文や while 文を考慮することが困難であるためであり，この結果，抽出される特徴である通信内容の順序には，実際にはキャプチャデータに現れないものや，実際には何度もキャプチャデータに現れるものがあると考えられる．このような抽出される通信の特徴の制限を考慮して，パケットの分類手法を検討する．

参考文献

[1] Xu, K., Zhang, Z.-L. and Bhattacharyya, S.: Profiling Internet Backbone Traffic: Behavior Models and Applications, *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, New York, NY, USA, ACM, pp. 169–180 (2005).

[2] Antoniades, D., Polychronakis, M., Antonatos, S., Markatos, E. P., Ubik, S. and Oslebo, A.: Appmon: An Application for Accurate per Application Network Traffic Characterisation, submitted for Broadband Europe (2006).

[3] Nguyen, T. T. and Armitage, G.: A Survey of Techniques for Internet Traffic Classification Using Machine Learning, *Commun. Surveys Tuts.*, Vol. 10, No. 4, pp. 56–76 (2008).

[4] Moore, A. W. and Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques, *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '05, New York, NY, USA, ACM, pp. 50–60 (2005).

[5] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A. and Salamatian, K.: Traffic Classification on the Fly, *SIGCOMM Comput. Commun. Rev.*, Vol. 36, No. 2, pp. 23–26 (2006).

[6] Sen, S., Spatscheck, O. and Wang, D.: Accurate, Scalable In-network Identification of P2P Traffic Using Application Signatures, *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, New York, NY, USA, ACM, pp. 512–521 (2004).

[7] : Network Log: https://play.google.com/store/apps/details?id=com.googlecode.networklog.

[8] Gibler, C., Crussell, J., Erickson, J. and Chen, H.: AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale, *Proceedings of the 5th International Conference on Trust and Trustworthy Computing*, TRUST'12, Berlin, Heidelberg, Springer-Verlag, pp. 291–307 (2012).

[9] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P. and Sheth, A. N.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, *ACM Trans. Comput. Syst.*, Vol. 32, No. 2, pp. 5:1–5:29 (2014).

[10] Dai, S., Tongaonkar, A., Wang, X., Nucci, A. and Song, D.: NetworkProfiler: Towards automatic fingerprinting of Android apps, *INFOCOM, 2013 Proceedings IEEE*, pp. 809–817 (2013).

[11] : Soot: http://sable.github.io/soot/.

[12] : Spika: http://spikaapp.com/.