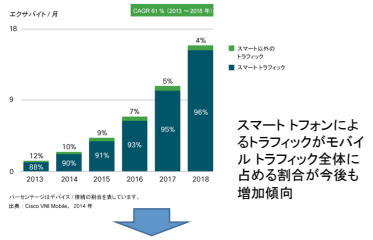
混在するスマートフォンアプリケーションのパケット分類手法

NTTネットワーク基盤技術研究所
中野雄介，上山憲昭，塩本公平
大阪大学大学院情報科学研究科
村田正幸，宮原秀夫
大阪大学サイバーメディアセンター
長谷川剛

背景

- スマートフォンの普及による，スマートフォンのトラフィックの増加



モバイルネットワークに対する影響の増大

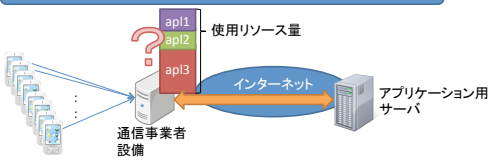
http://www.cisco.com/web/JP/solution/isp/ipngn/literature/white_paper_c11-520862.html

 2

課題

- 通信事業者以外の事業者やユーザによるアプリケーション作成
  - ネットワークへの負荷を意識しないアプリケーション

特定アプリケーションによるリソースの専有

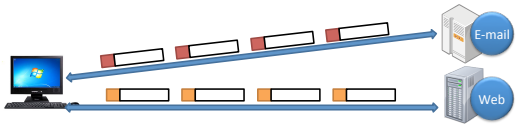


複数のスマートフォンアプリケーションのパケットが混在する
トラフィックデータをアプリケーションごとに解析をする必要がある

 3

既存技術

- TCPヘッダ，IPヘッダの情報でアプリケーションごとのパケットに分類



スマートフォンアプリケーションにはそのまま使えない

HTTPを用いるアプリケーションが多い

TCP,IPヘッダのみでアプリケーションごとに分類することが困難

 4

研究の目的

- 様々なスマートフォンアプリケーションのパケットが混在した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

将来的には

アプリケーションごとのトラフィック解析の実現

※Androidアプリケーションを対象とする

 5

提案手法

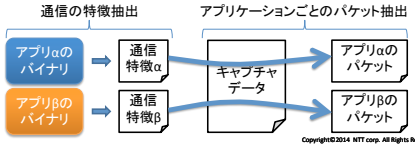
- 下記2つの解析手法を組み合わせる

Androidアプリケーションの通信の特徴を抽出

ソースコードに含まれる通信関連の文字列と宛先ホスト名とを通信の特徴として抽出

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抽出

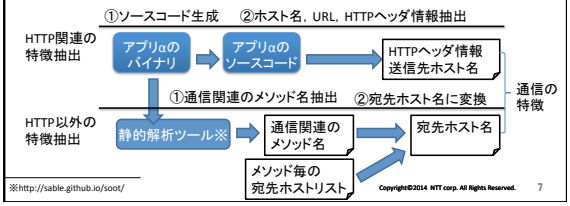
抽出された通信の特徴を用い，キャプチャデータから各アプリケーションの通信の特徴と類似するパケットをアプリケーションごとに抽出



 6

Androidアプリケーションの通信の特徴抽出

- HTTP関連の特徴抽出
  1. 各アプリケーションのAPKファイルからソースコードを生成
  2. ソースコードから，URL，HTTPヘッダ情報を抽出
- HTTP以外の特徴抽出
  1. 静的解析ツールで通信関連のメソッド名を抽出
  2. 予め作成されたメソッド毎の宛先ホストリストを用い，通信関連のメソッド名から宛先ホスト名に変換



Androidアプリケーションの通信の特徴抽出–詳細–

- HTTP関連の特徴
  - ソースコードからのキーワードに一致する部分を抽出
    - HttpPost
    - HttpGet
    - setHeader
    - addHeader
    - addRequestHeader
    - setRequestProperty
    - setRequestMethod
    - http://
    - https://

ソースコード中の記述例

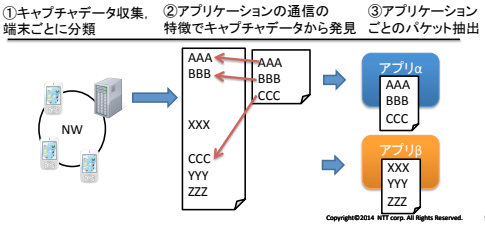
```
localHttpGet.setHeader("Content-Type", "application/json");
localHttpPost.setHeader("Content-Type", "application/
json");

public static final String BASE_URL = "http://
officialapi.spikaapp.com";
public static final String INFORMATION_URL = "http://
officialapi.spikaapp.com/page/information/";
public static final String LIST_SERVERS_URL = "http://
officialapi.spikaapp.com/api/servers";
```

- HTTP以外の特徴
  - メソッド名から宛先ホスト名に変換
    - GoogleCloudMessaging → android.googleapis.com
    - Ads → android.clients.google.com, googleads.g.doubleclick.net, redirector.gvt1.com, r5---sn-3pm7enes.gvt1.com, dl.google.com, api.crittercism.com, www.googleapis.com

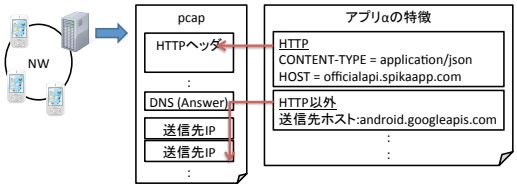
アプリケーションごとのパケット抽出

1. スマートフォンアプリケーションのパケットを含む通信をキャプチャし，端末のIPアドレス毎に分類
2. 通信の特徴を用い，キャプチャデータから各アプリの通信の特徴と一致するパケットを発見
3. 発見されたパケットを各アプリケーションのパケットとして抽出



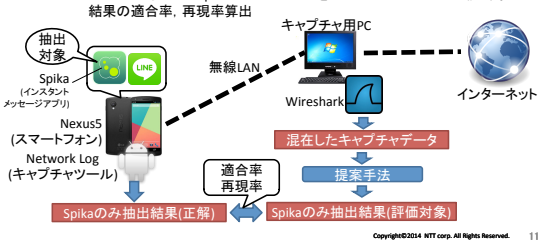
アプリケーションごとのパケット抽出–詳細–

- HTTP
  - 各アプリケーション特有のHTTPヘッダと一致するパケットをキャプチャ結果から発見
- HTTP以外
  - 送信先ホスト名に対するDNSクエリを発見し，そのAnswerのIPアドレスを送信先とするパケットをキャプチャ結果から発見



提案手法の評価

- 評価手法
  1. Spika，Line（インスタントメッセージアプリ）が混在したキャプチャデータ取得
     1. Spika:メッセージ送信　Line:アイドル状態
  2. キャプチャデータから提案手法を用い，Spikaのみのパケット抽出
  3. スマートフォン内でSpikaのみのパケットをキャプチャしたものと比較し，抽出結果の適合率，再現率算出



提案手法の評価

- 評価結果

適合率	再現率
0.98075	1.0

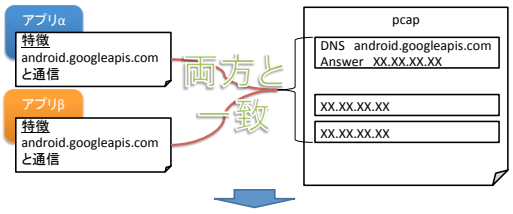
  - 適合率：抽出されたSpikaのパケット数/抽出されたパケット数
  - 再現率：抽出されたSpikaのパケット数/抽出するべきSpikaの全パケット数
- 考察
  - 誤って抽出したと判定された理由(適合率)
    - Network LogはTLSのパケットを抽出対象としていなかった．一方，提案手法はTLSを抽出したため．
      ⇒提案手法が誤って抽出したわけではない．

今後，更に多くのAndroidアプリケーションを評価対象とする

今後の課題

複数のアプリケーションが同一のホストと通信する場合，
アプリケーションごとの特徴の違いが無く，パケットの分類が困難

複数アプリがGoogle Cloud Messagingと通信する例



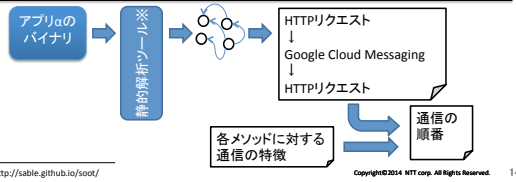
通信の順番の考慮が必要

今後の課題

-通信の順番を考慮した特徴の抽出-

1. 各アプリケーションのAPKファイルから静的解析ツール(Soot)によって制御フローグラフを生成
2. 制御フローグラフから通信関連のメソッドのみの呼び出し順番を生成
3. 各メソッドに対する通信の特徴(後述)と，通信関連のメソッドのみの制御フローグラフから通信の順番を生成

①制御フローグラフ抽出　②通信関連メソッドの呼び出し順番生成　③アプリケーションの通信の順番を生成

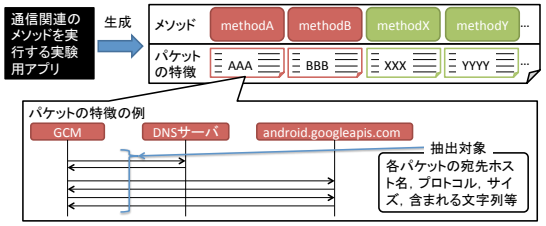


※http://sable.github.io/soot/

今後の課題

-各メソッドに対する通信の特徴の抽出-

1. 通信関連のメソッドを実行する実験用アプリケーションを作成
2. 実験用アプリケーションを実行することで，各メソッドに対する通信の特徴を抽出

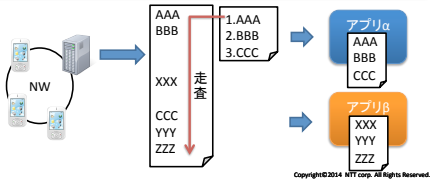


今後の課題

-通信の順番を考慮したパケット抽出-

1. スマートフォンアプリケーションのパケットを含む通信をキャプチャし，端末のIPアドレス毎に分類
2. 通信の順番を用い，キャプチャデータを走査し，各アプリの順番と一致するパケットを発見
3. 発見されたパケットを各アプリケーションのパケットとして抽出

①キャプチャデータ収集，端末ごとに分類　②アプリケーションの通信の特徴でキャプチャデータを走査　③アプリケーションごとのパケット抽出



まとめ

- 混在するスマートフォンアプリケーションのパケット分類手法を提案
  - アプリケーションごとに，ソースコードに含まれる通信関連の文字列を通信の特徴として抽出
  - 抽出された通信の特徴を用い，キャプチャデータから各アプリケーションの通信の特徴と類似するパケットを抽出
  - Spikaのパケットのみを分類できていることを確認
- 今後は，
  - 通信の順番を考慮することで，共通のホストと通信する複数のアプリケーションのパケットを分類できる手法を実現
  - 評価対象のアプリケーションの増加

