Vol.2016-IOT-32 No.33
2016/3/4

ブロードキャストパケットを用いたネットワーク接続機器の分別手法の提案

福田 直也[1,a)]　森 真幸[2,b)]　桝田 秀夫[2,c)]

概要：大学や企業などの組織において安定したネットワークを提供するため，接続されているデバイスを把握することは重要である．しかし，ユーザがネットワークに接続するデバイスは多種多様であり，それらが起因となるトラブルには，機種や OS の特定から始めなければならず，対応に苦慮する場合がある．この問題を解決するために，ネットワークに接続されたデバイスの OS を，ブロードキャストパケットを用いて，プライバシーに配慮して，分別する手法を提案する．

Proposal of a Method to Classify the Configuration of Devices in Network using Broadcast Packets

Naoya Fukuda[1,a)]　Masayuki Mori[2,b)]　Hideo Masuda[2,c)]

Abstract: Management of connected devices is essential to provide stability in the network at universities and company organizations. User's devices with various types can connect to the network easily. However, when trouble occurs in the network resulting from one of these devices, the process of determining the problem has to begin with identification of configurations such as device type and OS of the device causing the problem and this results in maintenances to be more difficult. Therefore, in this study, I proposed a method to firstly classify the configurations of devices in the network using broadcast packets while putting privacy into consideration.

1. はじめに

インターネットの普及に伴い，大学や企業などの組織において，コンピュータネットワークは組織の ICT (Information and Communication Technology) インフラとして様々な場面で利用されており，安定した運用管理の重要性が高まっている． 組織内で運用されているネットワークは， Layer2 スイッチやルータなどのネットワーク機器を組み合わせて構成されている．組織内ネットワークの管理者にとって，ネットワーク内に存在するデバイスの情報やネットワーク機器同士の接続状況はトラブル発生時に大いに役立つ重要な情報である．サポート期限が切れていたり，古いバージョンのままとなっている OS を使用してネットワークに接続するユーザは少なからず存在し，脆弱性を狙った攻撃を受けるだけでなく，ときには DDoS 攻撃に利用されるなど，何らかのネットワークトラブルを起こす恐れがある．このようなデバイスを放置すると，ネットワークの安定した運用を妨げたり，組織の信用にも関わるため，ネットワーク管理者にとって早急に解決すべき重大な問題となる．したがって，ネットワーク管理者はデバイスの情報を把握し，ユーザに対して効果的な指示が出せる必要がある．

企業などでは，ICT インフラの安定的な運用管理を実現するために，デバイスの接続に制約を持たせる場合がある．例えば，接続可能なデバイスを自社で用意したデバイスのみに限定したり，社外からの持ち込みデバイスは OS の種

[1] 京都工芸繊維大学 大学院 工芸科学研究科 情報工学専攻
Graduate School of Information Science，Kyoto Institute of Technology
[2] 京都工芸繊維大学 情報科学センター
Center for Information Science，Kyoto Institute of Technology
[a)] n-fukd15@dsm.cis.kit.ac.jp
[b)] m-mori@kit.ac.jp
[c)] h-masuda@kit.ac.jp

類やセキュリティ状態などの非常に厳密な条件を満たさなければ接続を禁止するといった運用がある．また，検疫ネットワークを構成し，デバイスに対してセキュリティ状態などを検査し，一定の検査に合格すれば社内ネットワークに接続できるようにするといった運用も見られる．検疫ネットワークを構成する場合，多くの実装では，接続しようとするデバイスに専用のアプリケーション (エージェント) をインストールし，エージェントがデバイスのセキュリティ状態などを確認し，管理サーバにデバイスの検査結果などの情報を提示する．管理サーバは，セキュリティ状態が一定基準を満たしていると判断されたデバイスのみを社内 LAN に接続できるようにする．

大学のような教育機関では，このような厳密な管理を行っているところは少ない．なぜなら，学生全員のデバイスを大学が用意するのは難しく，また，接続可能なデバイスを大学で制限することも難しいからである．そのため，学生が所有する様々な種類のデバイスを大学に持ち込むこととなる．学生が所有するデバイスを持ち込むたびに，そのデバイスが大学の提示する条件を満たしているかを調べるのは困難であり，デバイスにエージェントをインストールしておく運用方法も，様々な種類のデバイスが持ち込まれる大学のネットワークには適用し難い．また，卒業，入学によって一年ごとに一定数の学生が入れ替わるため，使用するデバイスも入れ替わるので，すべての変化を把握するのはネットワーク管理者にとって大変難しい．

近年ネットワークに既に接続されているデバイスに加えて，今までネットワークに接続しなかったデバイスにも通信機能を持たせる IoT (Internet of Things) という考えが広まっている．ネットワークに接続されるデバイスの種類は今後も増加することが予想される．しかし，そのようなデバイスの特徴を，ネットワークに接続される前に取得することは大変難しい．

そこで本研究では，ネットワークに接続しているデバイスが出すブロードキャストパケットを用いて，デバイスで動作する OS や機器名を分別する手法を提案する．本手法により，接続されたデバイスの情報を区分できるため，トラブルの切り分けやサポート可能な OS の選定などに活用できると考えらえる．また，各 OS 専用のソフトを必要としないエージェントレス型であり，トラブルの度にデバイス自身の調査をする手間も少なくなるため，管理者の負担軽減も期待される．

2. 関連研究

文献 [1] では，学外向けの HTTP 通信のログ情報を，発信元および，送信先 IP，URL，User Agent 情報を含んだ形で取得し記録している．Web ブラウザなどから発生する HTTP リクエストに含まれる User Agent 情報には，ブラウザ名，バージョン，OS の情報 などが含まれており，解析することで発信元の PC で利用している OS などの情報を推定できる可能性がある．

3. 検討

3.1 OS の分別に使用するパケット

ネットワーク通信は，通信相手の数により 3 種類に分類される．

- ユニキャスト ・・・ 1 : 1
  - 指定した相手にデータを送信
- マルチキャスト ・・・ 1 : 複数
  - 指定した相手 (複数) にデータを送信
- ブロードキャスト ・・・ 1 : 全ての LAN 接続機器
  - LAN 内に存在するすべてのホストに対して，同時にデータを送信

このうち，関連研究 [1] では，HTTP 通信を利用して OS を推定しているので，ユニキャストパケットを使用している．

以下の理由から，ブロードキャストパケットを使用して，デバイスの OS を分別する手法を提案する．

(1) 近年 HTTP 通信において HTTPS 通信を利用しようとする傾向があり，今後，ユニキャストパケットを使用した推定が可能か不透明である．
(2) ブロードキャストパケットはほとんど暗号化されていない．
(3) 暗号化されているデータを解析でき，利用できたとしても，暗号化されているデータまでみると通信の秘密に抵触する可能性がある．

ブロードキャストパケットは，DHCP[2]，ARP[3]，Bonjour，WINS[4] などのプロトコルで利用される．Bonjour では，主に MacOS で利用され，WINS は，主に Windows で利用される．DHCP 利用環境でデバイスをネットワークに接続する際は，OS に関係なく，すべてのデバイスが DHCP パケットを利用し IP アドレスを付与されることに着目し，提案手法では DHCP パケットを用いる．

3.2 DHCP

DHCP とは，Dynamic Host Configuration Protocol の略である．インターネットに一時的に接続するコンピュータに IP アドレスなど必要な情報を自動的に割り当てるプロトコルである．デバイスがネットワークに接続し，ネットワーク内に存在する DHCP サーバと情報をやりとりすることにより IP アドレスが付与され，インターネットにアクセス可能となる．このやり取りをする際に，デバイスが出力する DHCP Discover，DHCP Request パケットを解析する．

3.2.1 オプション

DHCP パケットのオプション部分は，可変長でありデバイスによって異なる．このオプションは，Hostname，

Vendor Class Identifier や Client ID などの文字列を含むパラメータ，Message Type や Parameter Request List などの数字のみのパラメータからなる．

3.2.2 Parameter Request List

Parameter Request List とは，DHCP サーバにより IP アドレスを付与される際に，デバイスが要求するネットワーク情報のリストである． Subnet Mask，Default Gateway，Domain Name Server などのパラメータからなる．

3.2.3 Vendor Class Identifier

Vendor Class Identifier とは，デバイスが，DHCP サーバに対して，特定の種類のハードウェアとファームウェアを使用していることを伝えるために設定する，可変長の文字列，またはオクテットである．DHCP サーバに登録されていれば，デバイスからの要求は適切に処理される．

4. クラスタリング

クラスタリングとは，分類対象の集合を，内的結合と外的分離が達せされるような部分集合に分割することであり，機械学習の一種で教師なし学習に分類される．分類対象のデータから，性質が近いもの同士をひとまとまりにする．このまとまりをクラスタという．

4.1 クラスタリング手法

デバイスのデータを分別する際に用いるクラスタリングソフトを，python の機械学習ライブラリである scikit learn[5] に決定した．このソフトは，クラスタリング，分類，回帰，次元削減などの機能を持つ．クラスタリングをする際の手法を，分割最適化手法の Affinity Propagation とした．これは，入力したデータから自動でクラスタ数を決定するものである．今回のように，観測するデバイスの種類がどのくらいあるのかが不明なため，クラスタ数をあらかじめ決定してクラスタリングを行うのは困難である．

4.2 文字列のクラスタリング

文字列の値をそのままクラスタリングするのは困難である．そこで，よく用いられる手法は，文字列をベクトル化し，クラスタリングを行うことである．その際に文字列の分割が重要になる．例えば，internetandoperationtechnology という文字列を分割する際に，期待される分割方法は internet/and/operation/technology である．

ここで，一つの文字列分割アルゴリズムを作成する．このアルゴリズムとは，アルファベット文字列とアルファベット文字列以外を分割するという単純なものである．例えば，internet34and+-operation*3technology という文字列を分割すると， internet/34/and/+-/operation/*3/technology となる．この時，アルファベット文字列のみをクラスタリング対象とする．先の例では，internet， and， operation，technology がクラスタリング対象となる．クラスタリング対象となる文字列を単語と定義する．

以下のような手順でクラスタリングを行う．

(1) ある文字列の中にある特定の単語と，他の文字列の中にある同じ特定の単語が存在する場合，それらの文字列は同じクラスタに属する可能性がある
(2) すべての文字列に対して繰り返し (1) の検証を行う
(3) クラスタの大きさが最大になるようにクラスタを作成する

5. 提案手法

5.1 DHCP PRL 手法

DHCP パケットのオプション部分にある Paramter Request List を用いた分別手法を提案する．この手法では，特定のリストがある，または，ない，というデータを特徴として用いる．観測したデバイスごとにこの特徴を抽出する．4.1 節で述べた Affinity Propagation を用い，クラスタリングを行う．

5.2 DHCP bool opts 手法

DHCP パケットのオプション部分全体を使用する分別手法を提案する．この手法では 5.1 節と同様に，特定のオプションがある，または，ない，というデータを特徴として用い，5.1 節で使用したパラメータも同時に用いる．その際，DHCP パケットの Discover， Request パケットの中身は異なることも利用する．4.1 節で述べた Affinity Propagation を用い，クラスタリングを行う．

5.3 DHCP bool after string opts 手法

文字列を含んだパラメータを利用する分別手法を提案する．5.1 節，5.2 節における分別手法では，特定の値がある，または，ない，というデータを特徴として用いた．しかし DHCP のオプション部分では，文字列を含んだパラメータもあり同列に扱うのは困難である．そこで，この手法では，文字列の部分と，文字列でない部分と，二回に分けてクラスタリングを行うことにする．その手順を示す．

(1) Vendor Class Identifier の文字列を用いて，4.2 節の手法でクラスタリングを行う．
(2) (1) の結果より，各クラスタに対して 5.2 節の手法でクラスタリングを行う．

6. 考察

本章では，提案手法に対する結果を示し，考察する．使用したデータは，本学分散システム研究室内のネットワークをモニタリングしたものである．デバイスで動作している OS の情報をアンケートにより収集した．しかし，一部のデバイスにおいて，OS が判らなかったものがあったため，不明な OS は Unknown として記述している．

6.4 節には，あらかじめ分別する各 OS の特徴を入手し，

どのくらいの正誤が得られるものか，評価する．

6.1 DHCP PRL 手法に対する考察

5.1 節で示した手法を用いてクラスタリングを行った．今回の実験では，クラスタリングに使用するデータの範囲をあらかじめ決定した．Windows 以外の OS のバージョンはすべて同じものとみなしている．使用したデータを表 1 に，結果を表 2 に示す．

表 2 より ゲーム機以外は OS，機器名ごとに分別された．Windows 以外の OS についてはバージョンまで区別していないため不明であるが，バージョンまで考慮された分別は行われていないと考えられる．クラスタ番号 7 では，Nintendo と Playstation が同じクラスタに分別されており，この部分も区別されることが期待された．

ある程度の分別はできており，分別されたクラスタに対してもう一度クラスタリングを用いると，バージョンまで区別された結果が得られると考えられる．

6.2 DHCP bool opts 手法に対する考察

5.2 節で示した手法を用いてクラスタリングを行った．結果を表 3 に示す．

表 3 より，OS，機器名ごとにある程度分別された．しかし，例えばクラスタ番号 5 には Nintendo の他に ILO[6] が分別されている．

ある程度の分別は可能ではあるが，DHCP のオプション部分を使用したのみで，バージョンまでの分別は困難であると考えられる．

6.3 DHCP bool after string opts 手法に対する考察

5.3 節で示した手法を用いてクラスタリングを行った．Vendor Class Indentifier の値は 16 種類であった．4.2 節の手法でクラスタリングした結果クラスタ数は 12 となった．

5.3 節 (2) の手法で，各クラスタをさらにクラスタリングした結果の一部を表 4，5，6 に示す．表に示さないクラスタは，クラスタの中身が同じ OS，機器名しか入っていないので詳細は省いている．

手法 5.3 節 (1) を用いることにより，大まかな OS，機器名の分別ができた．

表 4 は，Vendor Class Identifier の値がなかったクラスタとなり，5.2 節の手法と同じことをしているため，結果も表 3 と似たようなものとなっている．

表 1 クラスタリングに使用したデータ

OS	数
iOS	4
MacOS	7
Windows7	2
Windows8	6
AndroidOS	6
openSUSE	2
Ubuntu	2
Nintendo 3DS	2
Playstation 3	1
Nintendo Wii	1
PXEBoot	1
ILO	1

表 2 5.1 節の手法の結果

クラスタ	OS，機器名
1	iOS
2	MacOS
3	Windows7
	Windows8
4	AndroidOS
5	openSUSE
6	Ubuntu
7	Nintendo 3DS
	Playstation3
	Nintendo Wii
8	PXEBoot
9	ILO

表 3 5.2 節の手法の結果

クラスタ	OS，機器名	数
1	Nintendo WiiU	1
	XBOX360	1
	Playstation Vita	1
	Playstation 3	1
	MR04LN	1
	Unknown	2
2	openSUSE Leap42．1	1
3	PXEBoot	1
4	Ubuntu	3
5	Nintendo Wii	1
	Nintendo 3DS	3
	ILO	2
6	AndroidOS4	9
	AndroidOS5	2
	AndroidOS6	1
	Unknown	1
7	CISCOAP	2
8	openSUSE 13．2	1
9	WindowsVista	1
	Windows7	6
	Windows8	5
	Windows10	5
10	Nintendo WiiU	1
	LGwebOSTV	2
	Unknown	1
11	iOS8	2
	iOS9	3
	MacOSX	2
	MR04LN	1
	Unknown	1

表 5 は，AndroidOS クラスタとなっている．クラスタ番号 3，6 以外はバージョンできれいに分別されているクラスタはない．そこから，AndroidOS4 との推定は可能であるが，AndroidOS5 の推定は困難であると考えらえる．また AndroidOS6 も使用したが，手法 5.3 節 (1) の結果により別クラスタに分別された．

表 6 は，Windows クラスタとなっている．すべての Windows が入るクラスタ，Vista，7 のクラスタ，8，10 のクラスタと分別されている．

Vendor Class Identifier の値があるものは，手法 5.3 節 (1) のクラスタリングが有効であると考えらえる．第二段階のクラスタリングがあまりうまく動いていないため，さらなる特徴の入手が必要と考えられる．

6.4 評価

OS の推定に 5.1 節と同様，DHCP Parameter Request List のみを用いる．この方法では，デバイスを推定する前に各 OS の特徴を入手しなければならない．したがって，手元にあるデバイスの種類が少なければ推定精度も下がる．

上記の方法を用いて評価を行う．表 7 に示した各 OS から，特徴を入手した．Windows 以外の OS では，バージョンの区別をなくし，すべて同じものとみなしている．この時，Ubuntu のみ複数のパラメータを出力したため，複数のデバイスで観測されたパラメータを Ubuntu の特徴として用いた．

表 4 5.3 節の手法の結果 (1)

クラスタ	OS，機器名	数
1	openSUSE Leap42．1	1
2	Nintendo WiiU	1
	MR04LN	1
3	Ubuntu	3
4	Nintendo Wii	1
	Nitendo WiiU	1
	Nintendo 3DS	3
5	iOS8	2
	iOS9	3
	MacOS10	2
	Unknown	1
6	openSUSE 13．2	1
7	LGwebOSTV	2

表 5 5.3 節の手法の結果 (2)

クラスタ	AndroidOS	数
1	4	6
	5	1
	Other	1
2	4	2
	5	1
	Other	1
3	4	3
4	4	7
	5	1
	Unknown	1
5	4	2
	5	1
6	4	2

表 6 5.3 節の手法の結果 (3)

クラスタ	Windows	数
1	Vista	1
	7	5
	8	5
	10	5
2	Vista	1
	7	5
3	8	5
	10	4

結果を表 8 に示す．表 8 の 1 列目は OS 名である．2 列目はデータを入力したときに，その OS と分別されたデバイスの数 である．3 列目は，分別された OS が正解だったデバイスの数である．4 列目は，分別された OS とは別のものだったデバイスの数である．この時，WindowsVista と Windows7 は同じ特徴だったため，区別ができないので同じ行にまとめてある．Unknown には，特徴を入手する際に選べれなかった OS や機器名が分別されている．たとえば，CentOS やゲーム機などは Unknown となる．

表 8 より，iOS，AndroidOS において，誤推定が発生した．原因は PXEBoot[7] などのパケットであった．その他の OS で誤推定は発生しなかった．

Paramter Request List のみを用いることである程度の分別が可能であることがわかった．しかし，この手法では OS を分別するためには，各 OS との特徴の違いを発見しなくてはならない．OS のメジャーアップデートや，新しく発売されたデバイスなど，今までと異なった新規の特徴を持ったものが出現することも考えられ，その都度データ収集をしなければいけない．また Windows についてはバージョンまで詳しく見ているため，正確に分別できていると考えられるが，他の OS については，バージョンまで分別可能かは不明である．

表 7 6.4 節のサンプル

OS	数
iOS	6
MacOS	6
WindowsXP	2
WindowsVista	3
Windows7	6
Windows8	8
AndroidOS	10
openSUSE	2
Ubuntu	3

表 8　6.4 節の結果

OS	推定数	正	誤
iOS	10	8	2
MacOS	7	7	0
WindowsXP	2	2	0
WindowsVista，7	11	11	0
Windows8	10	10	0
AndroidOS	13	11	2
openSUSE	3	3	0
Ubuntu	12	12	0
Unknown	15	0	0

7. おわりに

デバイスで動作する OS，機器名の分別手法を提案した．どの提案手法も，OS，機器名レベルでの分別は可能であることがわかった．しかし，OS のバージョンまでの分別をするには情報が不足していることがわかった．6.4 節より，あらかじめ各 OS の特徴を入手することによる推定は，特徴を調査するコストがかかり難しい．今後，新規のデバイスがネットワークに接続してきた際にも適応でき，他のブロードキャストパケットを用いた分別手法と組み合わした手法で，分別精度をあげることが必要と考えられる．

参考文献

[1] 鳩野逸生：HTTP 通信ログ解析による学内情報機器の利用状況推定，インターネットと運用技術シンポジウム 2014，pp. 63–70 (2014).
[2] Droms, R.: Dynamic Host Configuration Protocol, http://www.ietf.org/rfc/rfc2131 (1997).
[3] Plummer, D. C.: An Ethernet Address Resolution Protocol, http://www.ietf.org/rfc/rfc826 (1982).
[4] Microsoft: WINS defined: Windows Internet Name Service (WINS), https://technet.microsoft.com/en-us/library/cc784707 (参照日. 2016/02/02).
[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E.: Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830 (2011).
[6] Enterprise, H. P.: Integrated Lights-Out テクノロジー (iLO)，http://50146.www5.hp.com/products/servers/proliant/essentials/ilo-adv_sh.html (参照日. 2016/02/02).
[7] Henry, M. and Johnston, M.: Preboot Execution Environment (PXE) Specification, http://www.pix.net/software/pxeboot/archive/pxespec.pdf (1999).