「マルチメディア，分散，協調とモバイル
(DICOMO2019) シンポジウム」 令和元年7月

深層学習を用いた複数端末の無線LAN通信時のパケット解析と輻輳の予測

山本 葵[1]　山口 実靖[2]　神山 剛[3]　小口 正人[1]

概要：近年，世界中に増え続けているスマートフォン，タブレット端末は機能や性能も強化されている．気軽にネットワークにアクセスし，動画やゲームなどのデータ通信を楽しむことが出来るようになり，大容量かつ高速な通信に対する需要は増大している．しかし有線接続に比べ低帯域かつノイズの多い無線接続においては，膨大なパケットが通信中に無線 LAN アクセスポイントに蓄積され，その結果輻輳が発生してしまうという問題も生じている．本研究では輻輳発生前に制御を加え無線 LAN AP の輻輳を回避することを最終目的とし，本稿では目的達成のため輻輳の予測を行う．Android 端末を用いて無線 LAN 通信を行い，アクセスポイント周りのパケットをキャプチャした．そのパケットを深層学習の LSTM モデルを用いて解析し無線 LAN 通信時のトラフィックの予測性能を評価した．

Congestion Prediction by Wi-Fi Packet Analysis of Multiple Terminals Using Deep Learning

AOI YAMAMOTO[1]　SANEYASU YAMAGUCHI[2]　TAKESHI KAMIYAMA[3]　MASATO OGUCHI[1]

1. はじめに

モバイルネットワークにアクセスするスマートフォンやタブレット端末などのワイヤレスデバイスは世界中で増加し続けている．毎年多様な形状のワイヤレスデバイスが市場に登場し，これらの機能や性能も強化され進化し続けている．全世界のワイヤレスデバイス数は，2020 年までには 116 億にまで増加し，1 人あたりのデバイス数は 1.5 台になるという予想もなされている [1]．

端末自体の高機能化，高性能化は気軽にホームページの閲覧や音楽，動画やゲームなどのデータ通信を行うことを容易にしている．このようなデータ通信は大容量になることも多くあり得る．大容量かつ高速な通信に対する需要は増大すると考えられ，全世界のモバイルトラフィック量は 2015 年から 2020 年の間に約 8 倍も増加すると予想されている．[1]．

このことから大量のデータ通信がワイヤレスデバイスから発生しており，電波帯域は圧迫され，足りなくなりつつある．この問題を解決しようと携帯電話事業者は基地局の設置とともにデータオフロード，その中でも Wi-Fi オフロードを推進している．混雑しているワイヤレスデバイス-基地局間の無線部分のデータ通信を，Wi-Fi 経由にすることでデータを有線接続の固定回線などに誘導し，無線部分の通信量を減らそうとしている．実際に街中の施設では Wi-Fi オフロードを推進している携帯電話事業者などが配布した AP が多く見られる．

このような背景から今後無線 LAN へアクセスする端末数やトラフィック量が増加し，ワイヤレスデバイス-アクセスポイント (AP) 間においても大量のデータ通信が発生し帯域が圧迫されることが考えられる．ワイヤレスデバイスから送信されるパケットを処理する AP の負荷が増大し，輻輳が起こる頻度も多くなると考えられる．

TCP プロトコルは，様々な目的でのデータ通信において標準的に利用されているプロトコルの 1 つであり，ネットワークの公平かつ効率的な利用のため、TCP には輻輳制御アルゴリズムが備わっている．Linux や Android が採用しているロスベースアルゴリズムはパケットのロスを観測し，ロスが増加すると輻輳が発生したとして送信量を抑

[1] お茶の水女子大学
[2] 工学院大学
[3] 九州大学

えるというアルゴリズムである．しかし，より高いスループットを得るためにアグレッシブにパケットを送信するため，有線接続に比べて低帯域かつノイズの多い無線接続環境においてはその手法によって膨大なパケットが蓄積されてしまうという問題が生じることがある．この問題の解決法としては，高速通信の規格化があり，使用可能な周波数帯の増加や伝送速度の向上があれば輻輳は起きにくくなる．しかし規格が広く普及するには時間がかかり，実際街中では狭い帯域を取り合っているのが現状である．

よって本研究では無線 LAN AP の輻輳を回避することを最終目的とし，本稿ではこの目的を達成するために輻輳を予測できるか検証する．輻輳の発生が事前に検知できれば，輻輳ウインドウ（CWND）の補正などによって，輻輳発生前にパケットの送信量を抑えることができる．その結果パケットロスを発生させることなく，また無線 LAN AP に接続している全端末が平等に安定したスループットを確保できる輻輳制御が実現できる可能性がある．

本稿では，無線通信のトラフィックを解析し，輻輳の極めて早期における検出，予兆の発見をし輻輳の予測を行う．具体的には，Android 端末を用いてデータ通信を行い，無線 LAN AP 周りのパケットをキャプチャデバイスを用いて取得し，入力データとする．また各 Android 端末において通信速度，スループットを測定して記録する．これらのデータを用いて深層学習を行い，トラフィックの予測が可能であるか検証した．

2. 関連研究

2.1 カーネルモニタ

カーネル内部の処理は通常バックグラウンドで進められているため，通常ユーザ空間からその処理の様子を監視することはできない．そこで先行研究 [2] によりカーネル内部の情報を見るためにカーネルモニタというツールが開発された．TCP のソースコードにモニタ関数を挿入し，カーネルを再構築することで，メモリからログを得ることができる．これにより輻輳ウインドウサイズや往復遅延時間 (RTT)，各種エラーイベントの発生タイミングなどの TCP パラメータをリアルタイムにモニタすることができる．

2.2 輻輳制御ミドルウェア

先行研究 [3][4] で開発された輻輳制御ミドルウェアは，カーネルモニタをベースとしたシステムであり，Android 端末間の連携した制御を目的としている．Android 端末間でこれから送信するセグメント数を表す輻輳ウインドウ値を通知し，周辺端末の通信状況を把握する．さらに周辺端末から受けた情報に基づき，輻輳ウインドウの上限値を自動で算出し補正することで，端末間で可用帯域を公平に分け合う．これにより無線 LAN AP に於ける ACK パケットの蓄積を回避する．

さらに [5] では輻輳制御ミドルウェアの改良を行っている．システムの発動タイミングを調整し多くの端末が同時に通信するときの全体の通信速度と公平性の向上を可能にした．

本研究はカーネルモニタで得ることができる情報を用いて，無線 LAN AP に接続される端末において輻輳を回避するために制御を行うことを目標としている点で関連研究を継承している．しかし輻輳が起こった後に輻輳を検知して制御を加える関連研究対し，本研究は輻輳が起こることを予想して輻輳が起こる前に制御を加えることを目標としている点で異なる．

3. 深層学習

深層学習はニューラルネットワークの階層を深めたアルゴリズムで，機械学習を実装するための 1 つの手法である．機械が自分自身で特徴量を抽出できるようになり，また階層を深めることで精度が大幅に向上した．これにより現在は第 3 次 AI ブームとも言われている．代表的な実用例は，郵便局で郵便番号を認識して選別する際に使われる文字認識，Amazon の売り上げを大きくあげたことで有名な商品レコメンドシステム，自動車の運転支援システムなど幅広く使われている．深層学習で用いられるモデルをさらに詳しく説明する．

3.1 ニューラルネットワーク (NN)

人間の脳内にある神経細胞とその繋がり，神経回路網を人工ニューロンという数式的なモデルで表現したもので，一般的なものは図 1 のように多数の層から一方向へ情報が伝搬されるようなモデルがよく使われる．層と層の間にはニューロン同士の繋がりの強さを表す重みがある．NN の発案はコンピュータが普及し始めた時と言われており，当時のコンピュータは非力で膨大な計算を行えるほどの容量や計算能力ではなかったが，近年のコンピュータのスペックの向上により深層学習が容易となった．

3.2 リカレントニューラルネットワーク (RNN)

RNN(図 2) は，入力データは互いに独立であると仮定されていた NN と違い，時系列の流れに意味を持つデータの予測や分類に用いられるモデルである．RNN は以前に計算された情報を覚えておくための記憶力を持っている．理論的には長いデータを記憶し，利用することが可能だが，実際は 2，3 ステップ前くらいの記憶しか維持できないという欠点がある．

3.3 ロングショートタームメモリネットワーク (LSTM)

LSTM(図 3) は RNN の拡張として登場した，時系列データに対するモデルである．LSTM は RNN の隠れ層のユニットを LSTM block と呼ばれるメモリと 3 つのゲートを

もつブロックに置き換えることで実現された．その最も大きな特徴は RNN ではできなかった長期依存が可能であるということである．

本研究は時系列データであるパケットの解析であるため，この LSTM をモデルとした深層学習を行う．

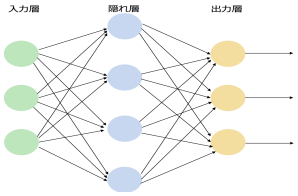

図 1　NN

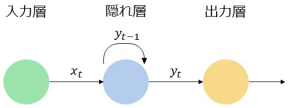

図 2　RNN

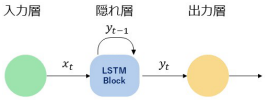

図 3　LSTM

4.　予測対象とする輻輳状態

本章では Android 端末から一斉にパケットを送信し急激にトラフィックを増加させることで，意図的に輻輳状態をつくりだす．輻輳状態で様々なパラメータの振る舞いを観察することにより，輻輳を示す値としてどのパラメータを採用すべきか実験を行い決定する．各パラメータを観察するために，先行研究 [2] にて開発されたカーネルモニタを導入した Android 端末を用いて通信を行い，通信時の TCP パラメータを取得する．また，通信には iperf を使用して通信速度を測定する．実験環境は図 4 である．輻輳の発生を観察するために 200 秒間パラメータを取得した．0-10 秒は Android 端末からは一切パケットを送信はせず，10 秒付近で全ての Android 端末から一斉に iperf によりサーバにパケットを送信を開始し，10-190 秒でパケットを送信し．190-200 秒はパケット送信をしない．全端末においてパケットを送信する，しないの極端なデータ通信を行い意図的に輻輳を発生させた．5 台の Android 端末から各々の CWND 値，RTT 値，通信速度と 3 つのパラメータを取得し，グラフとして出力したものが図 5-図 7 である．

4.1　合計通信速度と 1 秒あたりの入出力パケット

図 5 は 5 台同時通信時の合計通信速度と 1 秒あたりの入出力パケット数のグラフである．10-190 秒で Android 端末は通信を行なっているが，26-29 秒，77-78 秒，97-98 秒，107-112 秒間など合計通信速度が 0Mbits/sec になっている時間がある．この合計通信速度が 0Mbits/sec になっている時間はパケットを送信しているが届いていないのか，パケットを送信すらしていないのか確かめるために 1 秒あたりの入出力パケット数を出力してみると．合計通信速度が 0Mbits/sec の時間は入出力するパケットが非常に少ないことがわかる．よってこの時間はパケットを送信していないことがわかる．

4.2　5 台同時通信時の CWND

図 6 は Android 端末 5 台の輻輳ウインドウ (CWND) の振る舞いを測定した結果である．10 秒までは通信を行っていないので値は 0 で一定だが，パケットを送信し始めた 10 秒付近からは各端末が自由にパケットを送出しており，TCP が各端末を制御している様子が見て取れる．Android 端末からパケットを送信し始めた 10 秒付近以降，輻輳を検知した TCP が CWND の値を変えて制御を行っている．Android が採用している輻輳制御アルゴリズムであるロスベースアルゴリズムは，パケットロスの検知により輻輳と判断している．よって図 6 の CWND の振る舞いをみると，各端末が自由に大量のパケットを送出し，パケットロスが起こるまで指数関数的に CWND が増加している．その結果輻輳が発生し，パケットロスを検知した TCP が CWND の値を変更することによる制御が行われていることが確認できる．

また大部分の時間において 5 台の端末は同じ程度のパケットを送信しているが，時間によっては各端末ごとに

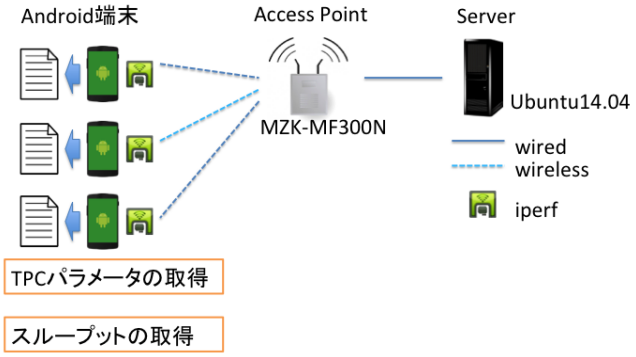

図 4　基礎実験環境

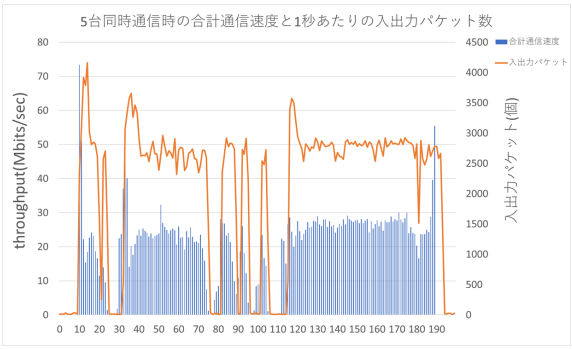

図 5　5 台同時通信における合計通信速度と 1 秒あたりの入出力パケット

CWND にばらつきがあることがわかる．

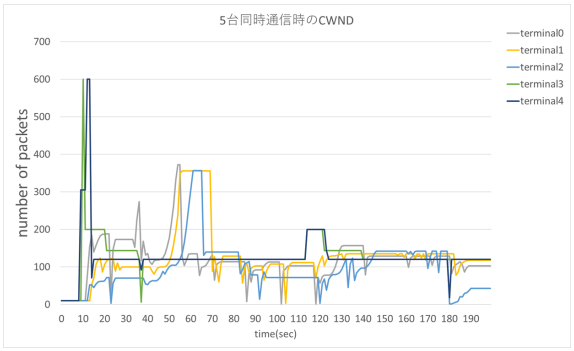

図 6　5 台同時通信における Android 端末の CWND の推移

4.3　5 台同時通信時の RTT 値

図 7 は Android 端末-Server 間の RTT 値の振る舞いを測定した結果である．

パケットを送信し始めた 10 秒付近から RTT 値の大幅な増加が見て取れる．これは 5 台の端末から大量にパケットが送られてくるため，キューが溜まる速度も速くパケットの処理が追いつかなかったことがわかる．また端末 3 と端末 4 の RTT 値はほとんど増加おらず，その他の端末の RTT 値が大幅に増加していることがわかり，5 台通信時において端末は平等に安定した通信が行われていないことがわかる．

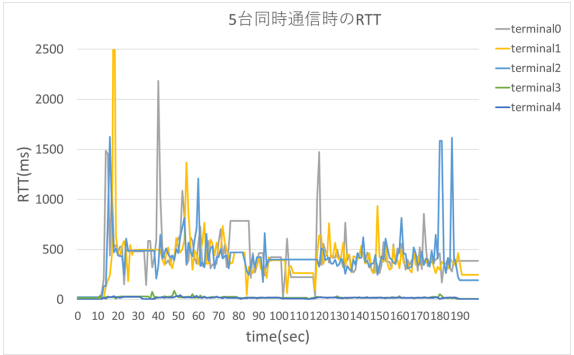

図 7　5 台同時通信における RTT 値の推移

4.4　輻輳を示す値 (予測する値)

実験により輻輳を示す値として合計通信速度を用いる．合計通信速度が 0Mbits/sec となっているときは AP が非常に混雑していると考えられ，また通信をしたいのにできていない状態というのはネットワークが正常ではないと考えらる．今後制御することで避けるべき状態であるので合計通信速度を深層学習の正解データとして予測する．よって次の章では通信時のパケット情報等から合計通信速度を予測できるか実験する．

5.　実験

本章では 4 章のように Android 端末がパケットを送信したくても送信できず，数秒間パケットの送信が行われなくなる輻輳状態を深層学習で事前に予測できるか評価実験を行なった

5.1　データセット

深層学習に用いるデータセットについて説明する．正解データには 4 章より合計通信速度を使用する．

5.1.1　入力データ

入力データの特徴量として無線 LAN AP 周辺のパケット情報を使用する．用いるキャプチャデバイスは米国，riverbed 社の AirPcap[6] である．AirPcap は wireshark 統合型のワイヤレストラフィックパケットキャプチャデバイスで，制御，管理，データの各フレームを含む，IEEE802.11 a/b/g/n のトラフィックをキャプチャできるデバイスである．AirPcap からキャプチャしたパケット情報は数値化し特徴量とする．さらに端末ごとにカーネルモニタから取得できるある時間 t 秒における CWND 値と RTT 値を変換し特徴量として入力データに加える．具体的には，ある時間 t 秒の端末ごとに取得できる CWND 値を合計し，端末数で割ることである時間 t 秒における平均 CWND 値を算出する．またある時間 t 秒の CWND 値は各端末によってばらつきがある．この t 秒における CWND 値のばらつきの大きさを特徴量としたいため，ある時間 t 秒における各端末の CWND 値の分散を算出し CWND 分散とする．RTT 値においても同様に平均と分散を算出し特徴量として入力データに加える．このようにして図 8 のような形の入力データと正解データのセットが．キャプチャしたデータの秒数分作成され，データセットとして学習やテストに用いられる．

	入力データ	正解データ
1秒間	[length sequence_number ... cwnd平均 cwnd分散 rtt平均 rtt分散] (1パケット+cwnd+rtt)	合計通信速度
	[length sequence_number ... cwnd平均 cwnd分散 rtt平均 rtt分散]	
	...	
	[length sequence_number ... cwnd平均 cwnd分散 rtt平均 rtt分散]	

図 8　データセット

5.2　深層学習用フレームワーク

今回の実験に用いた深層学習用フレームワークは Chaier である．これは Preferred Networks 社が開発し，2015 年に公開された Python のライブラリである．特徴として”Flexible”，”Intuitive”，”Powerful”を掲げている [7]．

5.3　実験環境

5.1 章のデータを取得し実験を行う．実験環境を図 9 に

示した．1 台の AP に接続した複数台の Android 端末から iperf[8] によって，データをサーバに送信する．それと同時に PC で上記のデータを取得する．その後，適切なデータ形式に加工し，LSTM モデルを用いた深層学習をおこなう．

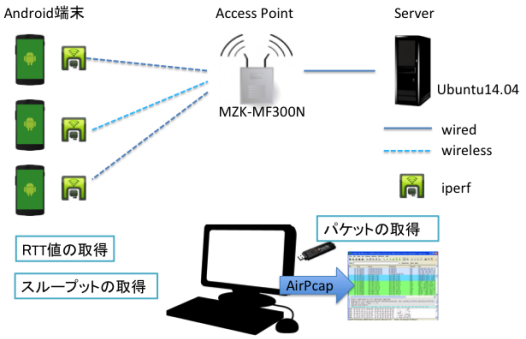

図 9　実験環境

表 1　実験機器の性能

Android	Model number	Nexus S	Nexus 7(2013)
	Firmware version	4.1.1	6.0.0
	CPU	1.0 GHz Cortex-A8	Quad-core 1.5 GHz Krait
	Memory(Internal)	16 GB，512 MB RAM	16 GB，2 GB RAM
	WLAN	Wi-Fi 802.11 b/g/n	Wi-Fi 802.11 a/b/g/n
server	OS	Ubuntu 14.04 (64bit) / Linux 3.13.0	
	CPU	Intel(R) Core(TM)2 Quad CPU Q8400	
	Main Memory	8.1GiB	
AP	Model	MZK-MF300N(Planex)	
	Support Format	IEEE 802.11 n/g/b	
	Channel	13	
	Frequency Band	2.4 GHz(2，1412-2，472 MHz)	

表 2　解析に用いた PC の性能

PC	OS	14.04.1-Ubuntu
	CPU	Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
	GPU	GeForce GTX 1080
	Memory	2.4 GHz(32GB)

5.4　学習

5 台通信時のデータから作成したデータセットを用いて学習を行なった．4 章と同じく 5 台の Android 端末を用いて通信を行い，200 秒間通信を行い通信時のデータを取得してデータセットとした．0-10 秒はパケットを送信せず，10-190 秒で iperf を用いて通信し，その後 190-200 秒はパケットを送信しない．この 200 秒間で Wireshark より取得したパケット情報，カーネルモニタより CWND 値,RTT 値，iperf より通信速度のデータを図 8 のような特徴量をもつデータセットとし t 秒の入力データから t 秒の合計通信速度を予測する学習を行った．

5.5　学習データによる予測結果

図 10 は 200 秒の学習用データを学習させたモデルに再び同じ 200 秒の学習用データを入力データとして与えたものである．オレンジの線が正解の RTT 値，青の線が予測した値である．

図 10 より精度よく学習ができていることがわかる．

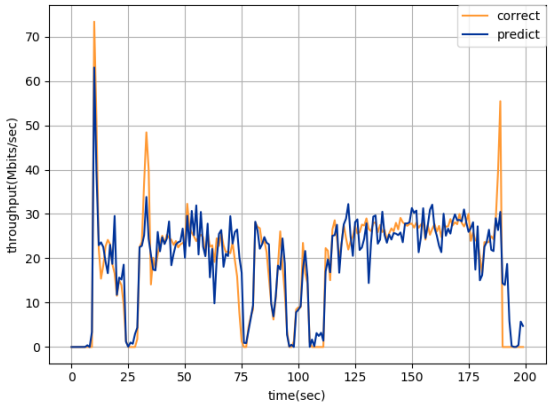

図 10　学習データによる予測結果)

5.6　テストデータによる予測結果

学習の汎化能力を検証するためテストデータによる検証を行う．テストデータは新規に作成する．7 台 Android 端末を用いて通信を行い，パケットの送信時間は学習データと同じようにし，200 秒間のデータを取得してデータセットとした．学習に用いた 200 秒のデータとは端末の台数が異なり，学習用データと異なるデータをモデルに入力することで汎化能力を検証する．

結果は図 11 となった．

図 11 は 7 台通信時のテスト用入力データを先ほど学習させたモデルに入力した結果である．全体的に値は正確に予測はできていないが，60 秒付近の合計通信速度が 0 に近づく箇所では，タイミング等は非常に良い精度で予測ができている．

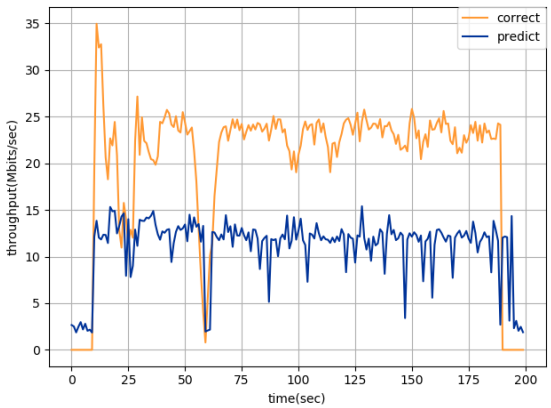

図 11　テストデータによる予測結果 (7 台通信)

5.7 数秒先の予測

5.4 章から 5.6 章での実験では t 秒までの入力データから t 秒の正解データである合計通信速度を予測できるか検証した．しかし制御を目的とした本研究では，制御するための計算時間を考慮しなくてはならないため，t 秒までの入力データから数秒先の正解データを予測し，輻輳を発見する必要がある．そこで 3 秒先を予測できるのか実験した．学習に用いたデータは 5.4 章の 5 台通信時の学習用データセットであり，t 秒までの入力データから t+3 秒の正解データを予測するように学習させる．

図 12 は学習用データを再度入力した結果である．3 秒先の予測では学習データによる予測結果はある程度は予測できているが，目的としている合計通信速度が 0Mbits/sec になる時間はほとんど予測できていない．数秒先を予測するには入力するデータ等さらに工夫が必要なことがわかった．

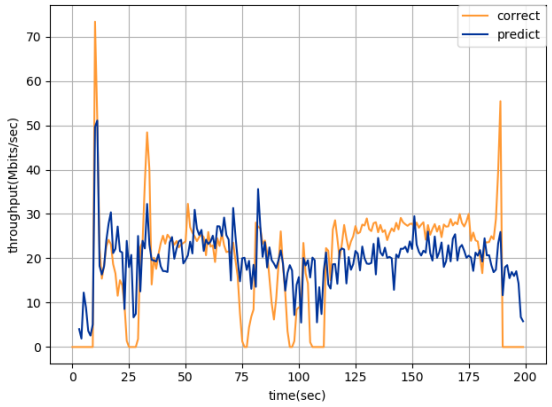

図 12 学習データによる 3 秒先の予測結果 (7 台通信)

6. まとめと今後の課題

本研究では，AP に接続する端末が複数台通信を行い，輻輳がおこるであろう場合において，輻輳発生前に制御を行うことを目標とし，その前段階として輻輳の極めて早期の検出，予兆の発見をするために深層学習を用いてトラフィックの予測を行なった．具体的には無線 LAN AP に接続した Android 端末を用いてデータ通信を行い，そのパケットをデバイスを用いてキャプチャする．そのパケット情報を数値化し，さらにカーネルモニタから取得した各 Android 端末の CWND 値を入力データとした．またカーネルモニタから得られる TCP パラメータのうち，各 Android 端末の CWND 値，RTT 値を，iperf より合計通信速度の振る舞いを観察することにより，合計通信速度が 0Mbits/sec になってしまう時間は避けるべき輻輳状態と考え，正解データを Android 端末の合計通信速度とし，深層学習を行なった．深層学習はフレームワークに Preferred Networks 社の Chainer を使い，時系列データに適する LSTM モデルを使用してトラフィックの予測が行えるか実験を行なった．

学習データによる予測は非常にうまくいっており，学習データの特徴を良く学習できていることがわかった．テストデータによる予測は全体的に正確な値が予測できず，改善の余地がある．

今後の課題としては様々な状況下でデータ通信を行なっている時の予測精度の向上が 1 番に挙げられる．数十分，数時間，数日とさらに長時間のデータを集めてあらゆる状況においてのデータを用いて十分に学習をさせていきたい．また端末の台数の変更や端末が一斉に通信を行なった場合だけでなく，ランダムに通信を行なっている場合においても精度よく予測ができるように学習を行って行きたい．

また今後輻輳制御を行っていくことを見据え，輻輳を発見してから制御を行うための計算時間も考慮に入れて実験を行っていきたい．いくら予測精度が良くても，予測するために計算している間に結果の予測時刻を過ぎてしまっては意味がない．どの程度先の時刻まで精度よく予測できるか，また数秒先の予測の精度をあげるため実験を行なっていきたい．

謝辞

本研究は一部，JST CREST JPMJCR1503 の支援を受けたものである．

参考文献

[1] Cisco Visual Networking Index, https://www.cisco.com/c/ja_jp/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html
[2] Kaori Miki, Saneyasu Yamaguchi, and Masato Oguchi: “Kernel Monitor of Transport Layer Developed for Android Working on Mobile Phone Terminals,” Proc. ICN2011, pp.297-302, January 2011.
[3] Hiromi Hirai, Saneyasu Yamaguchi, and Masato Oguchi: ”A Proposal on Cooperative Transmission Control Middleware on a Smartphone in a WLAN Environment,”Proc. IEEE WiMob2013, pp.710-717, October 2013.
[4] Ai Hayakawa, Saneyasu Yamaguchi, Masato Oguchi: “Reducing the TCP ACK Packet Backlog at the WLAN Access Point,” Proc. ACM IMCOM2015, 5-4, January 2015.
[5] Ayumi Shimada and Masato Oguchi: “A Study of Android Tables Performance,” Proc.DEIM2017,H2-3,March 2017
[6] riverbed, https://www.riverbed.com
[7] Chainer,Framework for Neural Networks, https://chainer.org/
[8] Iperf For Android Project in Distributed Systems, http://www.cs.technion.ac.il/ sakogan/DSL/2011/projects/iperf/index.html