Vol.2020-DPS-182 No.64
Vol.2020-CSEC-88 No.64
2020/3/13

深層学習を用いた無線LAN通信時のパケットの移動平均の解析に基づくスループットの予測による輻輳の事前発見

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，通信時の TCP パラメータや通信速度を測定する．その測定データを深層学習の 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 Based on Wi-Fi Packet Moving Average Analysis Using Deep Learning

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

[1] お茶の水女子大学
[2] 工学院大学
[3] 九州大学

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 に接続される端末の通信時に輻輳を回避するための制御を行うことを目標としている点で先行研究を継承している．しかし先行研究 [5] では，「同一ネットワーク内の少なくとも 1 台の RTT 値の大幅な上昇」をきっかけとして制御を行う．RTT 値の大幅な上昇はネットワークの混雑の指標として用いられており，そのため先行研究では少なくとも 1 度はネットワークは混雑している．その後に制御を行う手順である．それに対し本研究では輻輳が起こることを事前に予想するため，予測がうまくいけば，輻輳が起こる前に制御を加えることが可能となる．

輻輳が起こった後に輻輳を検知して制御を加える先行研究に対し，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．

Vol.2020-DPS-182 No.64
Vol.2020-CSEC-88 No.64
2020/3/13

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．

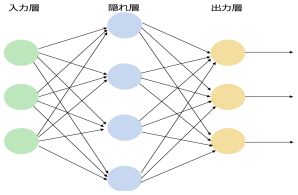

図 1　NN

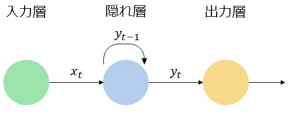

図 2　RNN

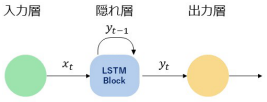

図 3　LSTM

4. 無線 LAN 通信においての TCP パラメータと通信速度のふるまい

本章では，無線 LAN 通信において端末の TCP パラメータと通信速度のふるまいについて調査を行った．

4.1 実験概要

本研究での実験環境下で，各端末の TCP パラメータと通信速度のふるまいについて観察を行った．

図 4 と表 1 に実験環境を示す．各 Android 端末には先行研究 [2] で開発されたカーネルモニタが導入されている．この環境において一つの AP に接続した Android 端末とサーバ間において iperf[8] を用いて通信を行い，ネットワーク全体の通信速度を測定した．カーネルモニタより各 Android 端末の TCP パラメータである CWND 値と RTT 値を，iperf により通信速度を得る．また通信中飛び交うパケットを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ある AirPcap[6] で取得し，パケット情報も得る．

実験開始から 10 秒で全ての Android 端末から一斉にパケットを送信し，その後 3590 秒の通信を連続的に行い．合計 3600 秒のパラメータを得た．尚，外界からの電波による妨害を防ぐため，クライアントと AP 間の空間はシールド布で覆い，できるだけ電波を遮断する環境において実験を行った．

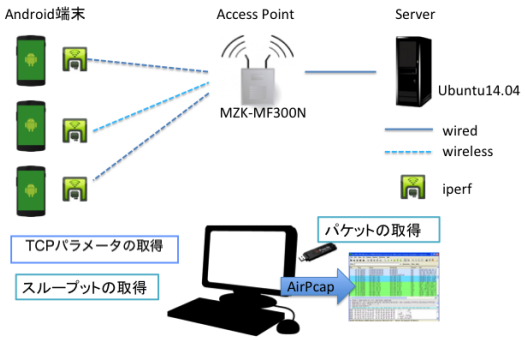

図 4　実験環境

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)	

4.2 CWND 値と合計通信速度の変化

3600 秒の通信で得られたパラメータ情報のうち，合計通信速度と cwnd 値の振る舞いについて 0-200 秒の結果を図 5 に，1770-2010 秒の結果を図 6 に示す．図 5 より 10 秒で一斉にパケットを送信したときは大きく cwnd 値があがっているが，その後は多少の値の増減はあるが，5 台の端末はほぼ同じような値を維持している．また合計通信速度も急激な変化はなく，5 台の端末が公平に通信を行なっていることがわかる．

その後安定した通信を行なっていたが，図 6 より 1790 秒付近で cwnd 値が大きく下がったあと，5 台ともほぼ同じ値を維持していた cwnd 値の均衡が崩れ，一部の端末のみが帯域を占有している様子がわかる．cwnd 値の大きな変化に伴い，合計通信速度も大きく増減し始めている様子がわかる．

4.3 RTT 値と合計通信速度の変化

3600 秒の通信で得られたパラメータ情報のうち，合計通信速度と RTT 値の振る舞いについて 0-200 秒の結果を図 7 に，1770-2010 秒の結果を図 8 に示す．図 7 より開始 10 秒での一斉にパケットを送信から，5 台の端末はほぼ同じような値を維持している．その後も安定して小さい RTT 値のまま通信をしていることがわかる．

その後図 8 より 1790 秒付近で cwnd 値や合計通信速度の値も大きく上下している時は，RTT 値も 1000ms を超えて急激に値を増減していることがわかる．

4.4 1 秒あたりのパケット数の変化

3600 秒の通信時に飛び交うパケットを AirPcap でキャプチャし，1 秒あたりにキャプチャすることのできたパケット数を算出した．得られたパケット数の変化について，0-200 秒の結果を図 9 に，1770-2010 秒の結果を図 10 に示す．図 9 より開始 10 秒での一斉パケットを送信時は，パケット数は急激に増加している．その後は多少の増減はあるものの，1 秒あたり 4000 個-5000 個の範囲内におおよそ収まっている．

その後図 10 より 1790 秒付近，他の TCP パラメータや合計通信速度の値も大きく上下している時は，飛び交うパケット数も大きく乱れ始め，安定してパケットを送信できていないことがわかる．

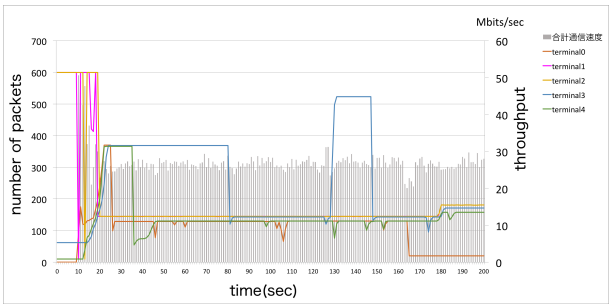

図 5　0-200 秒の合計通信速度と CWND 値の推移

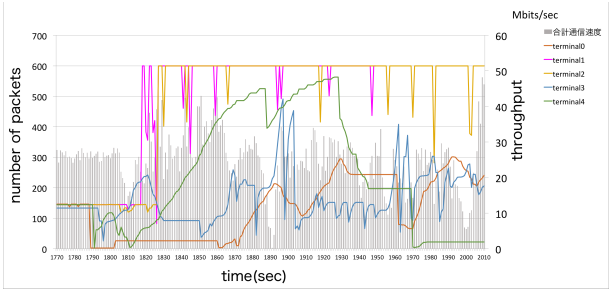

図 6　1770-2010 秒の合計通信速度と CWND 値の推移

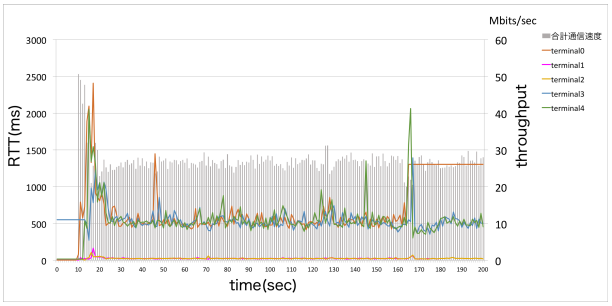

図 7　0-200 秒の合計通信速度と RTT 値の推移

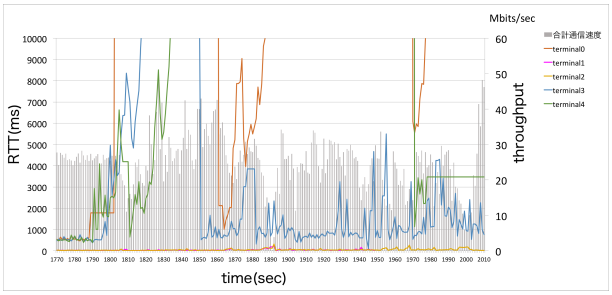

図 8　1770-2010 秒の合計通信速度と RTT 値の推移

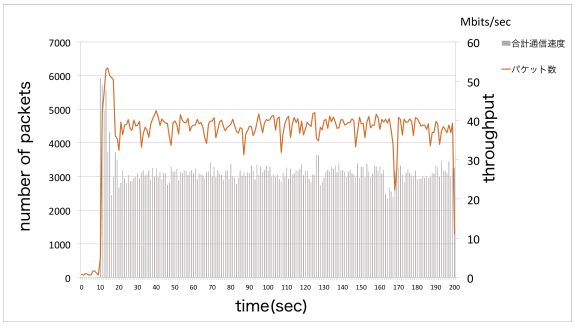

図 9　0-200 秒のパケット数の推移

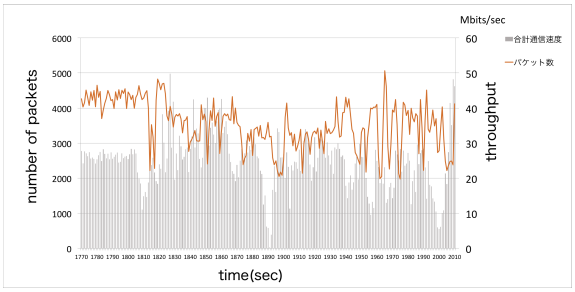

図 10　1770-2010 秒のパケット数の推移

5. 実験結果

以上の結果より，複数台の端末を用いて無線 LAN 通信を行うと，安定した状態と均衡が崩れ不安定な状態が確認できた．不安定な状態では CWND 値，RTT 値，合計通信速度，1 秒あたりにキャプチャできたパケット数において安定した状態に比べ，十分な値が確保できていないことがわかる．

この結果より合計通信速度が急激に減少するときはネットワークが不安定であり，AP が非常に混雑していると考察でき輻輳が発生していると考えられる．それに伴い Android 端末が TCP により輻輳制御を行うため，CWND 値が急激に変化している．一度不安定な状態に陥ると，端末それぞれが通信速度を確保するための TCP 制御を行うため，一部の端末のみが帯域を占有する状態も確認できる．一部の端末のみが帯域を占有し，各端末が十分な通信速度を出すことができない不安定な状態は，今後制御することで避けるべき状態であると考察することができる．

6. 予測実験

6.1 実験概要

前章で得られた通信時のデータをもとに予測実験を行う．

深層学習には時系列データに対するモデルである LSTM モデルを用いる．正解データには合計通信速度を用いる．合計通信速度を予測することは輻輳発生を予測することとなる．学習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．

学習に用いたデータは第 5 章で示した 3600 秒のデータのうち，不安定状態 (1790 秒-3600 秒) である 2000 秒-3000 秒時のデータを用いる．テストデータは学習データと同じく 3600 秒のデータのうち，3000 秒-3400 秒時のデータを用いる．t 秒までの入力データから t+1 秒の正解データを予測する．データセットは表 2 で示すものとする．その結果を図 11 と 12 に示す．図は特定の 200 秒のみ表示している．

オレンジの線が正解の合計通信速度，青の線が予測した値である．

表 2 データセット

入力データ	正解データ
合計通信速度	合計通信速度

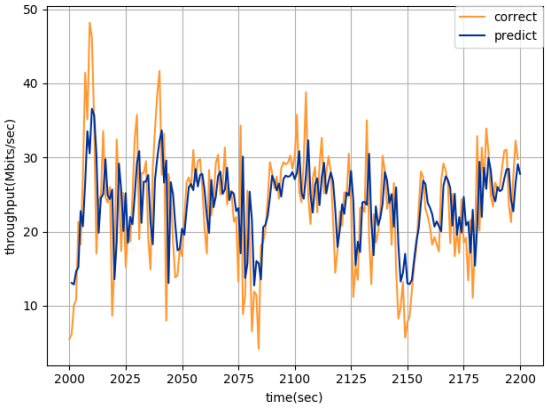

図 11 予測実験　結果 (学習データ)

6.2 結果考察

図 11 と 12 の結果より，ある程度精度よく学習が行われていることがわかる．しかし測定した合計通信速度は瞬間的な増減が多くみられ，合計通信速度が急減，急増するところは予測もうまくいっていない．全体的な合計通信速度の増減をとらえ輻輳を予測したい本研究では合計通信速度の急減急増はノイズとなってしまう可能性がある．そのためある程度合計通信速度をなだらかにし，ノイズを抑えるため移動平均をとることとする．

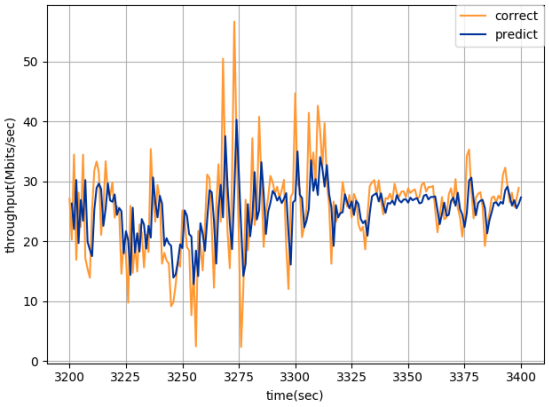

図 12 予測実験　結果 (テストデータ)

6.3 移動平均を用いた予測実験

移動平均を用いてなだらかにした合計通信速度を図 13 に示す．図 13 は 3600 秒のデータを用いて，t-4 から t 秒までの移動平均をとって算出した合計通信速度である．オレンジの線が測定したオリジナルの合計通信速度，青の線が移動平均を用いて算出した合計通信速度である．急減急増が抑えられている様子がわかる．

このように移動平均を用いて新たに算出した合計通信速度を用いて学習を行い予測をする．予測結果を図 14 と 15 に示す．オレンジの線が正解の合計通信速度，青の線が予測した値である．図 11 と 12 の結果と比較すると，視覚的にも精度が向上していることがわかる．移動平均を用いることで合計通信速度のノイズを抑えることができ，学習をうまく行わせることが可能となった．

しかし精度よく学習できているように見えるが，図を見ると予測が正解から少しずれており．1 秒前の入力結果を出力するように学習されているように見えることがわかる．

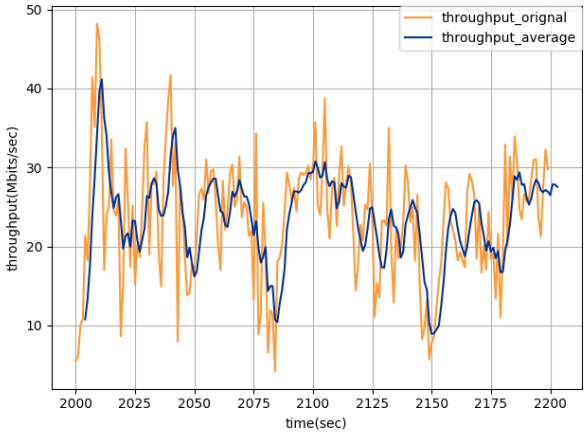

図 13 オリジナルの合計通信速度と移動平均を用いて算出した合計通信速度 (2000-2200 秒)

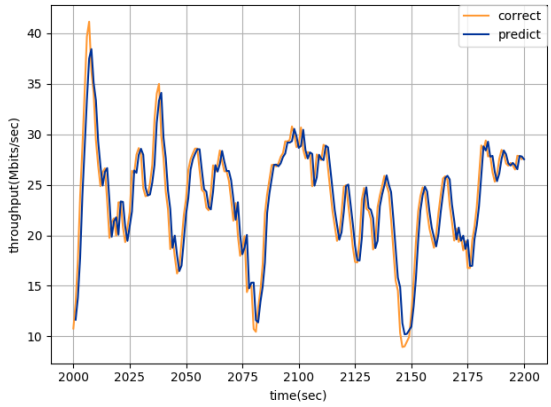

図 14　移動平均を用いた予測結果 (学習データ)

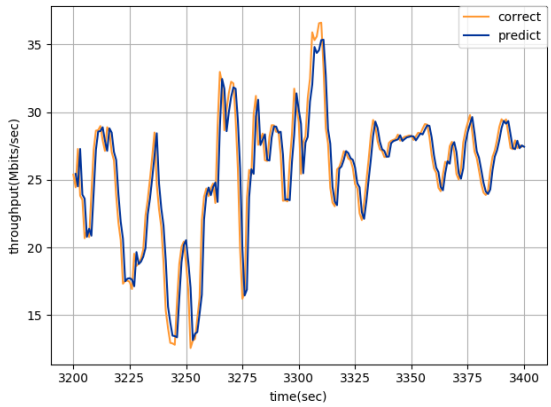

図 15　移動平均を用いた予測結果 (テストデータ)

6.4　移動平均を用いた予測結果と repeat 手法の誤差比較

移動平均を用いた予測は 1 秒前の入力結果を出力するように学習しているように見えるため，repeat 手法と誤差を比較することで評価を行う．repeat 手法とはひとつ前の結果をそのまま出力する手法であり，t+1 秒の予測結果がほしければ，t 秒の入力を出力する．

テストデータによる予測結果と式 1 の N=2 の場合を用いて N 乗誤差平均を算出する．算出した結果を図 16 に示す．移動平均を用いた予測の方が誤差が少ないことがわかり，repeat 手法より移動平均を用いた予測の方が優れていることがわかる．さらに N を 2 から 3 に増加させると，予測誤差と repeat 誤差の差が広がる．これは移動平均を用いた予測結果の方が repeat 手法による予測結果と比較して大きな予測はずしをしていないということである．

$$\text{N 乗誤差平均} = average(|correct_t - predict_t|^N) \quad (1)$$

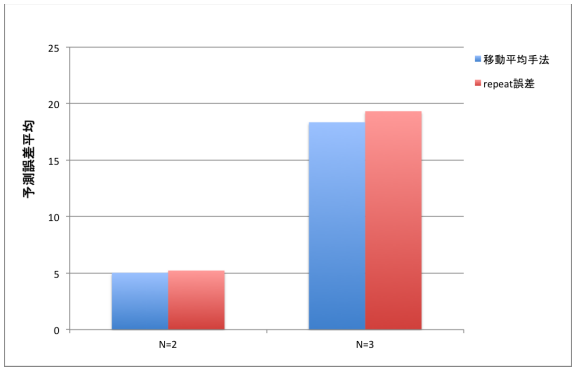

図 16　提案手法と repeat 手法による誤差の比較

6.5　数秒先の予測における移動平均を用いた予測と repeat 手法の誤差比較

本研究では輻輳状態の制御を最終目的としており，その前段階として輻輳を予測する．制御を加えるための計算時間を考慮し，予測を行わなければならない．そこで，1 秒先だけでなく，移動平均を用いて数秒先の予測を行い，repeat 誤差との比較を行う．テストデータによる予測結果と式 2 を用いて 2 乗誤差平均を求める．結果を図 17 に示す．2 秒先，3 秒先，4 秒先において移動平均を用いた予測の方が誤差が少ない．精度の向上が難しかったテストデータを用いた予測において，移動平均を用いた予測は repeat 手法を用いるよりもよい精度があることが確認できる．

$$\text{2 乗誤差平均} = average(|correct_t - predict_t|^2) \quad (2)$$

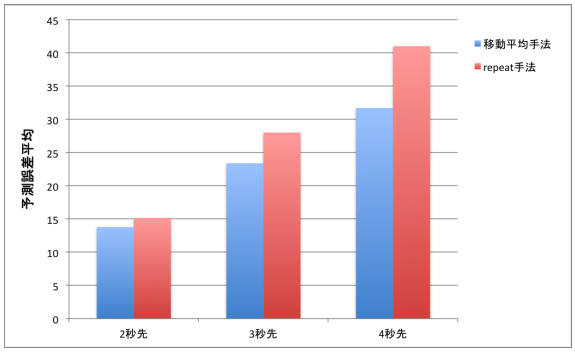

図 17　移動平均を用いた数秒先の予測と repeat 手法による誤差の比較

7.　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，iperf より合計通信速度の振る舞いを観察することにより，無線 LAN 通信には安定状態と不安定な状態があることがわかった．不安定な状態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．

本研究では，深層学習による予測が repeat 手法より優れていることを証明したが，予測精度のさらなる向上は次段階の制御を加えるタイミングと直結し，有用性を増すため精度の向上は大きな課題である．

また制御を前提とすると他にも様々な課題が存在する．移動平均を用いた予測を repeat 手法と比較して評価を行い，優れていることを示したが repeat 手法よりはよい精度に留まり，さらなる精度向上を目指さなくてはならない．先行研究よりもよいタイミングで制御をくわえるためには，一度もネットワークを混雑させてはならない．非常に良い精度の予測が必要となるため，学習モデルやデータセットについて引き続き調査し，数時間，数日に及ぶデータ取得と学習にも着手していきたい．また制御を目的としているため本来は予測にかかる計算時間，さらに制御を加えるまでの時間等を考慮し．どれくらい前に輻輳が発見できれば制御に間に合うのか明らかにする必要がある．

また今回の実験では，データ取得，深層学習と様々な機器を用いた．しかし実社会では，無線 LAN AP 周辺に大きな設備は設けられない．そのため手軽な方法でデータ取得からデータセットとし学習モデルに入力するまでの工程を実現しなければならない．またその後予測した輻輳発生タイミングを無線 LAN に接続している端末にフィードバックし，制御を行う方法については議論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