モバイルネットワークにおける周辺端末からの情報に基づく協調制御ミドルウェアの提案と実装

平井 弘実　(指導教員：小口 正人)

1　はじめに

近年スマートフォンの爆発的な普及と高性能化に伴い，スマートフォンを用いた大容量高速通信に対する需要が高くなっている．本研究は，このような需要に対応するため，スマートフォンに最適化した新しい通信制御の仕組みを提案する．

現行の輻輳制御アルゴリズムは，有線通信指向で開発されており，各端末が独立して，送出するセグメント数を見積もっていたため，ノイズの多い無線通信においては，最適な制御を行っているとは言い難く，課題が多く残されていた．そこで本稿では，アクセスポイント (AP) を共有している周辺端末を連携させ，互いの情報を利用して，より精密な可用帯域の見積りを行うミドルウェアを実装し，評価する．具体的には，周辺端末からの通信状況に関する情報通知に基づきネットワークの状態を判断し，各端末の輻輳制御を状況に応じて補正するといった仕組みである．本手法は，スマートフォンのネットワークトポロジを最適化する一提案であるが，今回はシェア率が最も高い Android 携帯を用いて実装を行い，有効性を実証する．

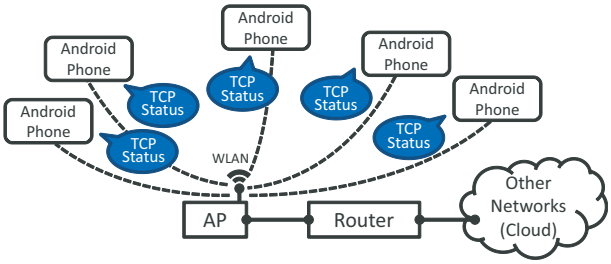

図 1: スマートフォンのネットワークトポロジ

2　協調制御ミドルウェアの提案

スマートフォンは主にクラウドによって支えられるサービスを利用しているため，通信のボトルネックとなる部分は，スマートフォン・AP 間の無線区間であると言える．既存の TCP はエンド・エンドのパケット損失率を頼りに輻輳制御を行っているが，スマートフォンのネットワークトポロジにおいては，AP を共有する端末が連携して全体のトラフィックを調整することは，通信スループットを向上させるために有効であると考えられる．本研究では，図 1 に示すように，AP を共有している周辺端末が協調し，連携した通信制御を行うことで，効率の良い通信制御を可能とするミドルウェアを提案する．

2.1　輻輳制御の補正

輻輳ウィンドウの値は通信速度と相関性があるため，輻輳ウィンドウの値を操作することで，通信速度が間接的に操作できる．しかし，本研究の評価環境においても有線経路で輻輳が起こらない保証はないため，フレキシブルにウィンドウスケールする必要がある．そこで我々は，無線の可用帯域を上回るパケット送出を防止し，同時に経路全体の輻輳を回避するためには，輻輳ウィンドウの最大値を設けることが望ましいと判断した．本稿では，適切な帯域活用を行うために輻輳ウィンドウの上限値を設定する本手法を「輻輳制御の補正」と呼ぶ．輻輳ウィンドウはカーネル内部の処理であるため，一般的にはユーザ空間からの入出力を受け付けない．そこで，本研究では Android のシステム上にプロセスインタフェースを実装し，ミドルウェアから書込み可能とした．

帯域を埋めるのに理想的な輻輳ウィンドウの値は以下の計算で求めることができる．帯域幅遅延積 (BDP) とは，送信側が通信経路に流すことができる最大データ量を示し，IEEE802.11g の場合は，最高転送速度 (20Mbps) ×往復遅延時間である．通信経路を周辺端末と均等にシェアするために，ネットワークの BDP を端末数 N で割った値を各端末の可用 BDP とし，各端末の理想的な輻輳ウィンドウの値は，可用 BDP ÷ [1 セグメントあたりのデータ量 (1.5Kbyte)] で求めることができる．この計算で理論上の理想的な輻輳ウィンドウを求めることが可能であるが，現実世界における複数端末の帯域共有は必ずしも理想的ではないため，我々は経験測から BDP に 120Kbits 程度の余裕 α を持たせることが必要だと判断した．これらの計算方法を以下の式にまとめる．

$$Ideal\ cwnd = \frac{BDP - \alpha}{Segment\ size \times N} \quad (\alpha = 120Kbits)$$

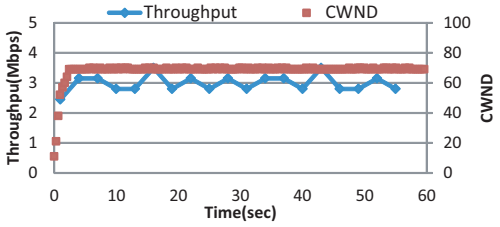

図 2: 補正した輻輳制御を持つ 5 台の端末で AP を共有した時の通信制御と通信速度の遷移

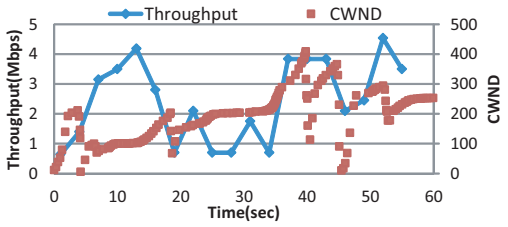

図 3: 標準の輻輳制御を利用する 5 台の端末で AP を共有した時の通信制御と通信速度の遷移

実際に，この式に基づき計算された理想的な輻輳ウィンドウの上限値を設定した端末を 5 台用意し，1 台の AP を共有して同時に通信を行った時の通信スループットと輻輳制御の振舞いを図 2 に示す．また同様の測定を標準の輻輳制御を利用する端末 5 台により行った時の様子を図 3 に示す．標準の TCP を利用した端末で

は輻輳ウィンドウは過度な増加と減少を繰り返しており，通信スループットもまた不安定となっている．一方で補正した輻輳制御においては，通信エラーがほとんど生じず，輻輳ウィンドウと通信スループットが安定した値を維持している．また端末数5〜8台で通信スループットを測定した結果を図4に示すが，最大2倍以上の速度の向上が確認できる．またこの時のFairness Indexを図5に示す．Fairness Index[1]とは，公平性を示す指標であり，1に近いほど高い公平性を示す．即ち，輻輳制御を補正し，理想的なパケット送出量を維持することで，転送エラーや再送を防ぎ，公平かつ効率的な通信が実現できたことがわかる．

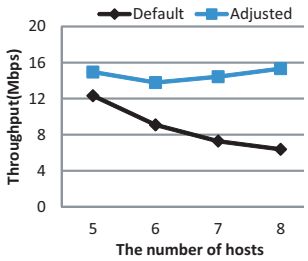

図 4: 通信速度の比較

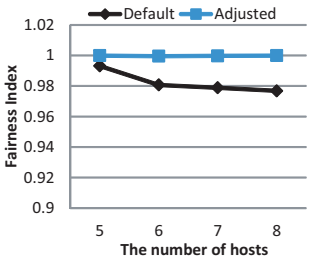

図 5: 公平性の比較

2.2 協調制御ミドルウェア実装

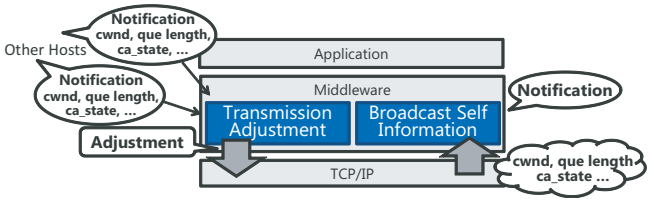

図 6: 協調制御ミドルウェアの概念図

協調制御ミドルウェアは，図6に示すように，周辺端末からの通知を受け，APを共有している端末数を考慮して，自端末の利用可能な帯域幅を計算し，その帯域幅を上回るパケットを送出しないよう，輻輳ウィンドウの上限値を設定する．

本ミドルウェアは，C言語で記述されたプログラムをクロスコンパイルすることで，Android端末に導入している．ネイティブバイナリは，Android独自のDalvikVMを利用しないため，バックグラウンドで端末に高負荷を与えることなく高速処理を行うことが可能である．また情報通知においては，UDPのブロードキャストを利用しているため，輻輳時には破棄され，オーバヘッドになる可能性は十分低い．

3 協調制御ミドルウェアの評価実験

3.1 実験概要

本ミドルウェアを評価するため，以下の実験を行った．実験では，図7に示すように，複数のAndroid端末が1台のAPを共有した状態で，Iperfを利用して同時にサーバ端末へデータ送信を行う．また高遅延環境を想定するため，サーバ機とAPの間に人工遅延装置Dummynetを設置し，人工遅延時間は往復で256msとする．最初に5台のAndroid端末は，240秒間のデータ送信を行うが，残りの3台のAndroid端末が，60秒おきに1台ずつデータ送信を始める．つまり最初の60秒間は端末数5台，次の60秒間は6台，そして最後の60秒間は8台へと送信端末数が変化する．この時，ミドルウェアを導入したAndroid端末は，周辺端末からの通知を受けて，アクティブにデータ送信を行っている端末数を考慮して輻輳制御を最適化する．また，本ミドルウェアの効果を明らかにするために，ミドルウェアを導入していない端末を8台用いた同様の実験を行い，比較対象とする．

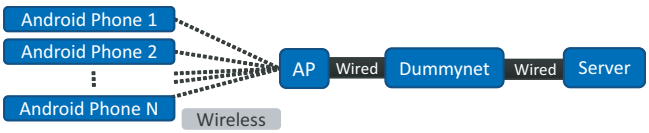

図 7: 実験装置

3.2 実験結果と考察

3.1節の実験結果を図8に示す．本実験は，ミドルウェアのオーバヘッドを含めた評価実験であるが，ミドルウェアを導入した端末は，ミドルウェアを導入していない端末における通信に比べ，通信スループットが向上していることが明らかである．既存の輻輳制御手法では，周辺端末数を考慮していないため，各々の端末が可用帯域に対して適切でない大きさのパケットを送出することにより，再送や通信エラーによる通信速度の低下やオーバフローが起きている．しかし，本ミドルウェアを導入した端末間においては，APを共有する端末同士が連携して，可用帯域を均等に分け合って使い切ることができたため，通信エラーや再送が減り，効率の良い通信制御が実現されたことが通信スループット向上の原因だと考えられる．

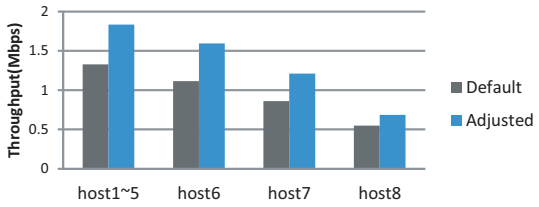

図 8: ミドルウェアによる通信性能向上

4 まとめと今後の課題

本稿では，モバイルネットワークにおける協調制御ミドルウェアの提案・実装を行い，その効果を示した．しかし，現実的な環境においては，様々なケースが考えられ，現在実装されているミドルウェアはその全てにおいて効果的とは限らない．例えばミドルウェアを持たない端末との協調や，低トラフィック通信を行っている端末との協調である．今後は，本ミドルウェアがより多くの環境で，効果的になるよう拡張を試みる．

参考文献

[1] D.-M. Chiu and R. Jain, “ Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, ” Computer Networks and ISDN Systems, vol. 17, pp. 1-14, 1989.