目次

概要

いつでも，どこからでも通信を行いたいという要望から，近年無線 LAN を通信インフラとして用いるサービスの需要が高まっている．無線 LAN の通信範囲を広げる方式として，アクセスポイント間を無線で結合することにより，バックボーンインフラを容易に構築するメッシュネットワークと呼ばれる方式に注目が集まっている．メッシュネットワークの実現例として，アクセスポイントがネットワーク内の全端末の情報を保持する方式があるが，この方式では端末数が増加したとき，制御パケットによるトラヒックの増加やテーブル量の増加が懸念される．

本論文では，通信要求発生時にオンデマンドで通信に必要なテーブルを生成する”WAPL（Wireless Access Point Link）”を提案する．WAPL では必要最小限のテーブルを保持するので，端末数が増加してもトラヒックを圧迫したり，テーブル量が増加することがない．また，WAPL は Ethernet を完全にエミュレートすることができ，Ethernet 環境で実現できる機能は全て実現可能である．よって，端末移動時におけるアクセスポイント間でのハンドオーバー等も容易に実現できる．

WAPL は，ルーティングプロトコルとは完全に独立した構造となっており，用途に応じてルーティングプロトコルの変更を行うことができる．本論文では，ネットワークシミュレータ ns2 を用い，様々な通信条件における WAPL の動作を解析し，最適なルーティングプロトコルを選定した．

第1章 はじめに

インターネットの急速な普及に伴い，いつでも，どこからでもインターネットへ接続できる無線 LAN の需要が高まってきている．無線 LAN エリアを広げるためにはアクセスポイント（Access Point 以後 AP）の整備が不可欠である．しかし，無線 LAN では AP 間を有線で結合するのが一般的であり，AP の設置には多大な工事費や時間を伴うのが現状である．また，一度設置した AP の移設や，新たな AP の増設が困難であるため，AP の計画的な配備が求められる．そこで，AP 間を無線で結合できれば，このような課題が解決され，無線エリアの拡大が容易になることが想定できる．

無線 LAN では，端末と AP 間はインフラストラクチャモードと呼ばれる方法で接続する．端末は近隣に存在する唯一の AP とアソシエーションを結び，必ず AP を経由した通信を行う．近年，新しいネットワーク形態として，アドホックネットワークが注目されている．アドホックネットワークとは，AP を必要とせず，通信端末のみで構成されるネットワークである．電波の届く範囲に端末が存在する場合，端末間で直接通信を行う．この方法をアドホックモードと呼ぶ．電波の届かない範囲に相手端末が存在する場合は，中間に位置するアドホックモードの端末がマルチホップでパケットを中継する．アドホックネットワークを形成するためには，端末どうしが相互に情報交換し，パケットの中継経路を決定する必要がある．これを実現するためのルーティングプロトコルは，IETF の MANET（Mobile Adhoc Network）ワーキンググループで標準化が行われている．アドホックネットワークではオールラウンドなルーティングプロトコルは存在しない．端末の移動速度，端末の存在密度などの状況に応じて最適なプロトコルが異なる．また，アドホックネットワークは他人の端末を無断で中継して電力を消費してしまうなどの問題があり，一般の用途への普及は進んでいないのが実情である．

そこで，アドホックネットワークを無線 LAN の AP 間を結合するためのプロトコルとして利用しようという試みがあり，メッシュネットワークと呼ばれて注目されている．メッシュネットワークでは，端末側はインフラストラクチャモードでよいとする方式が多い．この方法によると，インフラストラクチャモードで動作する端末に対して，無線 LAN のインフラを容易に確立できる．

メッシュネットワークの代表例として，Multihop-Wireless LAN（以後 M-WLAN）[1]-[4]，Metro Mesh[5]，MeshCruzer[6]といった技術がある．また，IEEE802.11s において標準化作業が進められようとしているが具体的な仕様は明らかになっている方式は少ない．ルーティングプロトコルとして，M-WLAN では MANET の

OLSR[7]，MeshCruzer では MANET の AODV[8]，Metro Mesh では独自の PWRP（Predictive Wireless Routing Protocol）を採用している．メッシュネットワークを用いて端末間通信を実現するためには，AP とその配下にある端末との関係を知る必要がある．M-WLAN では，OLSR を改良し，ネットワーク内の全ての AP が，全ての端末の物理アドレス情報を保持することによりこれを実現している．しかし，この方式では，端末数が増加すると，AP が管理するテーブル量が膨大になり，ルーティング情報を交換するための制御パケットのトラヒック量が増大するという課題がある．Metro Mesh，MeshCruzer に関しては，実現方式に関する詳細な情報はほとんど公開されていない．

そこで本論文では，メッシュネットワークを実現するための一方式として，端末間通信に必要なリンクテーブルをオンデマンドで生成することにより，テーブル量とトラヒックの増加を抑えることができる Wireless Access Point Link（以後 WAPL）を提案する[9][10]．WAPL では，端末からの通信要求があった時点で，通信に必要な最小限の情報のみを随時生成する．

WAPL はアドホックルーティングプロトコルとは完全に独立した構造となっており，ルーティングプロトコルをシステムに応じて入れ替えることが可能である．WAPL の応用として，例えば，災害発生により通信設備が破壊されたような状況下においても，通信環境を即座に回復させるために利用できると考えられる[11]．また，車車間通信に適用し，他車の搭乗者との間で P2P 通信を実現することも可能である[12]．

本論文では WAPL の実現方式を提案するとともに，シミュレーション解析により最適なアドホックルーティングプロトコルを調査した．適切な通信エリアのサイズを仮定し，ネットワーク内の通信ペア数の変化などが，通信トラヒックにどのような影響を与えるのかを，ネットワークシミュレータ ns2 を用いて検証した．その結果，WAPL には AODV が適していることがわかった．

以下，2 章で既存技術とその課題について述べる．3 章では WAPL の方式を，4 章では WAPL の構成と実装を，5 章ではシミュレーション結果と評価を述べ，6 章でまとめとする．

第2章 従来研究

本章では従来研究として，コンセプトが WAPL と似ており，かつ実現仕様が公開されている M-WLAN をとりあげ，その概要と課題について記述する．

第2.1節 M-WLAN

図 1 に M-WLAN のネットワーク構成図を示す．この図は WAPL にも適用される基本概念である．AP は無線 LAN のインタフェースを２つ持つ．一つは AP 間の通信用，もう一つはユーザ端末との通信を行う．AP 間通信はアドホックモードであり，MANET のルーティングプロトコルにより，AP 間の経路情報を生成する．ユーザ端末との通信はインフラストラクチャモードで行うため，ユーザ端末は一般の端末でよい．AP はユーザ端末のパケットをカプセル化/デカプセル化することにより，ユーザ端末間の通信を実現する．

図 2 に M-WLAN の動作概要を示す．AP は配下端末の情報（IP アドレス，MAC アドレス）を自身のアソシエーション情報としてアソシエーションテーブルに保持している．M-WLAN では，このアソシエーション情報を抜き出し，MANET のルーティングプロトコルでやり取りされる制御パケットの中に付加する．これにより，すべての AP がシステム内のすべての端末の情報をルーティングテーブルとして持つことができる. M-WLAN ではルーティングプロトコルとして Proactive 型の OLSR を採用する．Proactive 型とは，定期的に制御パケットのやり取りを行い，経路制御表を通信前に事前に作成しておく方式である．この制御パケット情報に，アソシエーション情報を追加することにより拡大ルーティングテーブルを生成する．

図 2 において，AP『A』と AP『D』が保持する拡大ルーティングテーブルの内容は図 3 の通りである．このように宛先アドレスとしてシステム内に存在する全 AP 及び端末の IP アドレスが格納される．次アドレスは次に送信すべき AP の IP アドレスを示す．

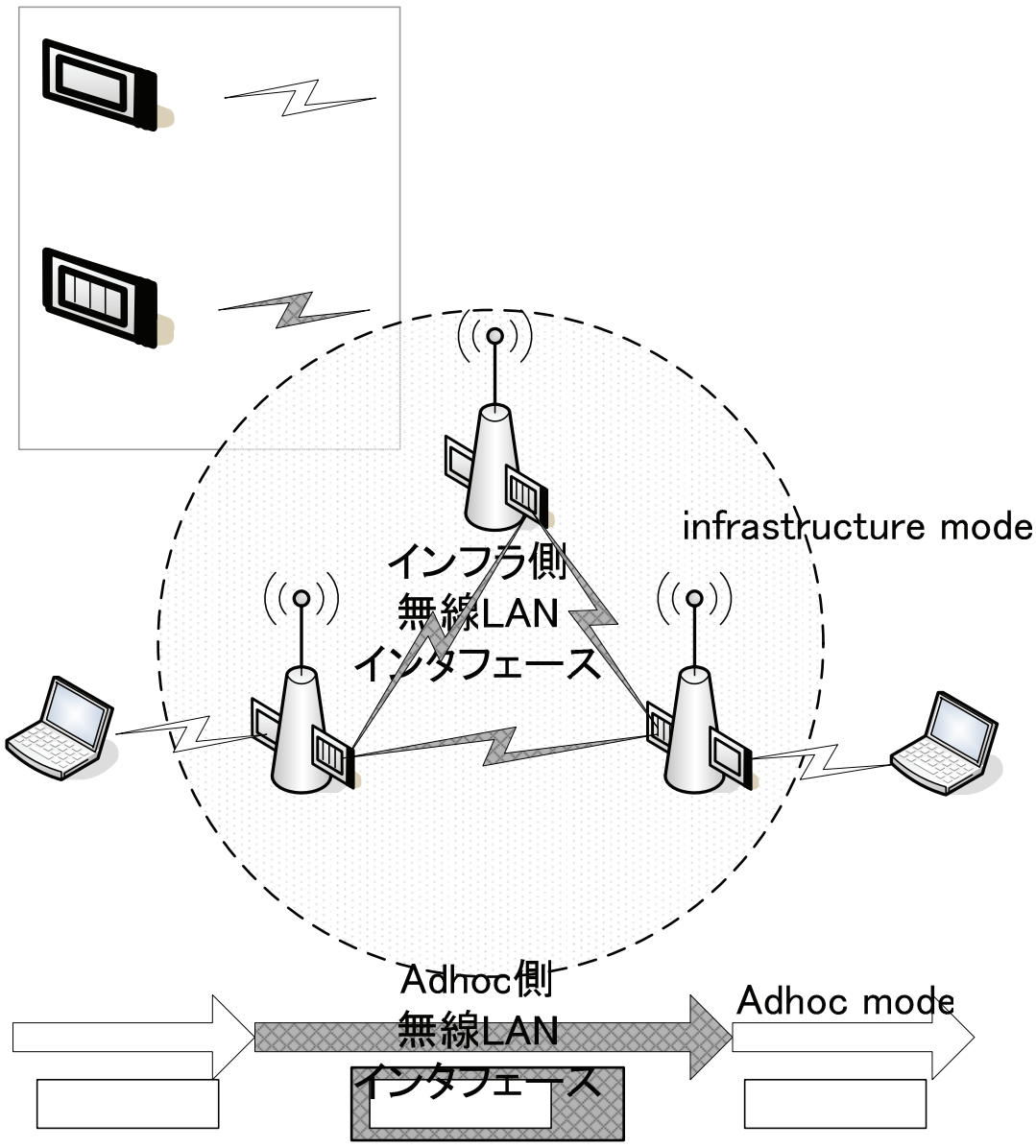


図 1　ネットワーク構成図

AP

端末

AP

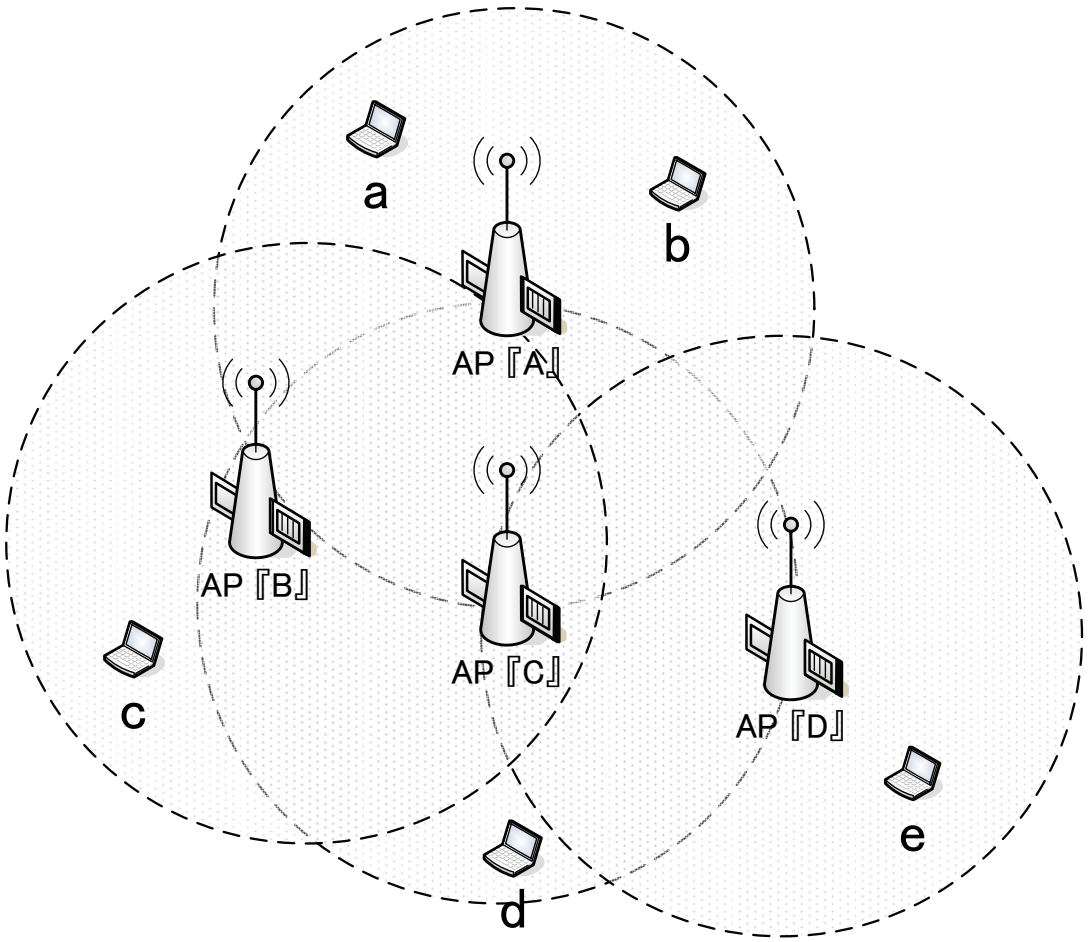


図 2　ネットワーク構成図

端末 a から端末 e まで通信を行うときの動作は以下の通りである．端末 a は自分の所属する AP『A』へパケットを送信する．AP『A』は受け取ったパケットの宛先アドレスが e であるため，拡大ルーティングテーブルを参照し，端末 e を配下に持つ AP『D』のアドレスでパケットをカプセル化する．カプセル化されたパケットは，ルーティングテーブルに従い，AP『D』までカプセル化されたまま届けられる．上記パケットを受け取った AP『D』は，カプセル化を解除し，宛先端末 e に送信する．

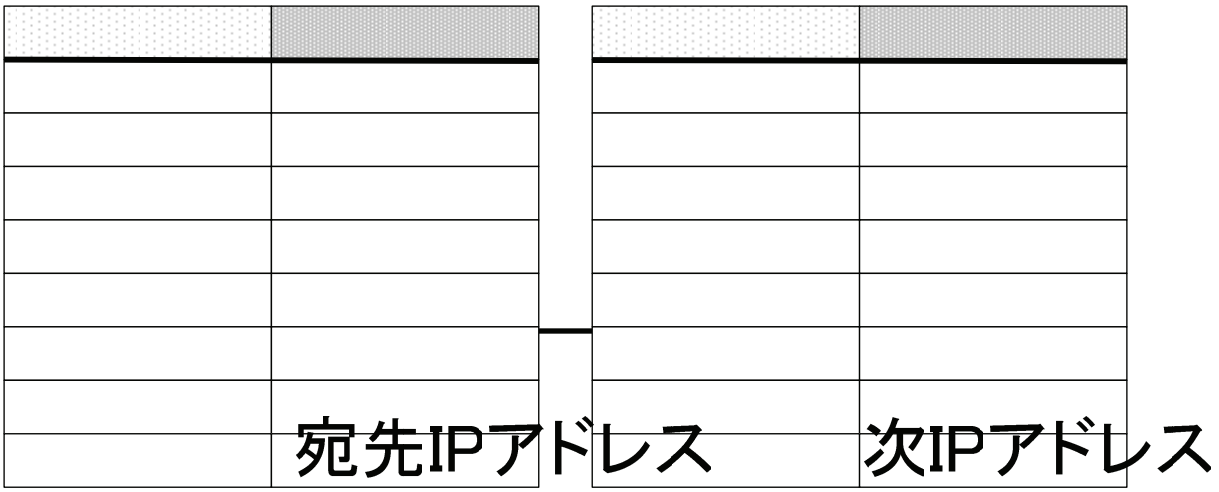


図 3　M-WLAN の保持する拡大ルーティングテーブル

端末のハンドオーバーについて説明する．端末の移動に伴い，アソシエーションが切り替わる．M-WLAN では，AP に新たに加わった端末からのフレームを常に監視している．AP はフレームを感知すると，自身のアソシエーションテーブルを変更し，ルーティングテーブルよりパケットのカプセル化を行う．このパケットを受け取った受信側 AP は，これまで受信していた端末のカプセル元 AP のアドレスが変わっているので，端末が移動したことを知り，ルーティングテーブルを書き換える．移動した端末がパケットを出さない場合は，アソシエーション情報を付加した制御パケットが定期的に送信されるタイミングで端末の移動を感知する．

M-WLAN における AP はアドホックネットワークのルーティングプロトコルを利用し，所持している情報すべて（端末の情報を含む）を制御パケットに乗せ，定期的に送受信する．したがって，小規模なネットワークでは大きな問題はないが，ネットワーク規模が大きくなると，AP で管理するルーティングテーブル量が膨大になる.このルーティングテーブルは定期的にフラッディングされるため，トラヒックへの影響が無視できない．また，端末の移動がルーティングテーブルに反映されるタイムラグは，アドホックプロトコルの定期パケットのタイミングに依存している．よって，移動した端末側が，パケットを送信しない場合，ルーティングテーブルの内容が再確定するまで通信が出来なくなる．

第3章 WAPL

第3.1節 動作概要

WAPL は，端末間の通信開始時に通信に必要なリンクテーブルをオンデマンドで生成する．WAPL の構成例を図 4 に示す．WAPL における AP を以後 WAP（Wireless Access Point）と呼ぶ．

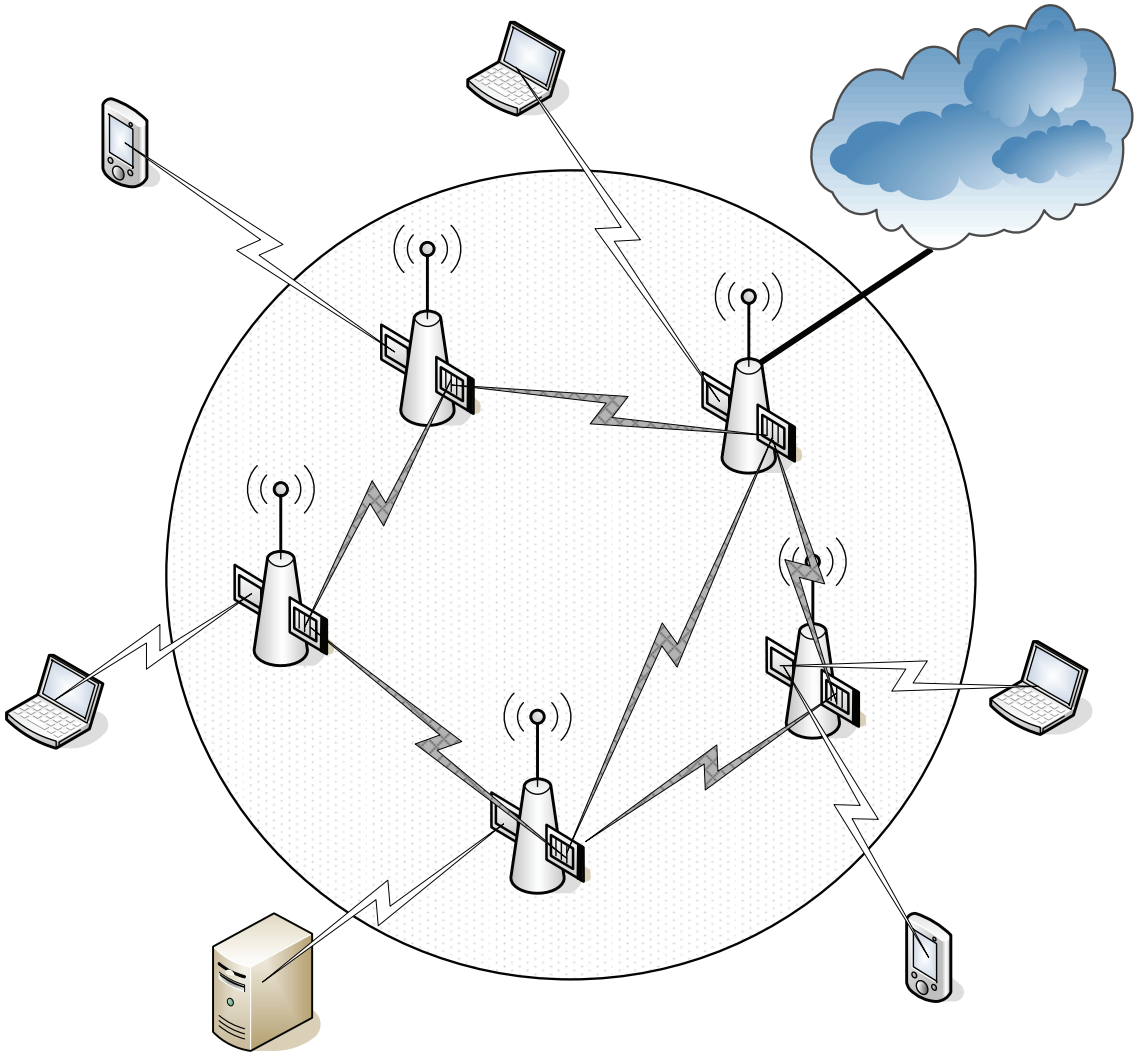


図 4 WAPL の構成例

WAP には無線 LAN インタフェースを 2 つ搭載し，端末－WAP 間はインフラストラクチャモード，WAP－WAP 間はアドホックモードで通信する．

WAPL では，アドホックルーティングテーブルとは別に，リンクテーブルと呼ばれるテーブルを独自に定義する．リンクテーブルとは，端末の MAC アドレスと従属する WAP のアドホック側 IP アドレスの対応関係を表すテーブルである．これを端末からの通信要求が発生した段階で，オンデマンドで自動生成する．通信に必要な最小限のテーブルを生成するため，無駄なメモリやトラヒックを削減することが可能となる．

WAP 間通信は MANET のルーティングプロトコルを使用するが，プロトコルには一切手を加えない．それゆえに，自在なプロトコルの選択が可能であり，用途に応じた最適なルーティングプロトコルを選択できる．また，無線エリアに接続する端末数が増加しても， WAP 間の制御トラヒックの量は変わらない．

第3.2節　リンクテーブルの生成

リンクテーブルの生成トリガーには，通信開始時に必ず実行されるアドレス解決プロトコル ARP（Address Resolution Protocol）[13]を利用する．図 5 にリンクテーブル生成シーケンスを示す．

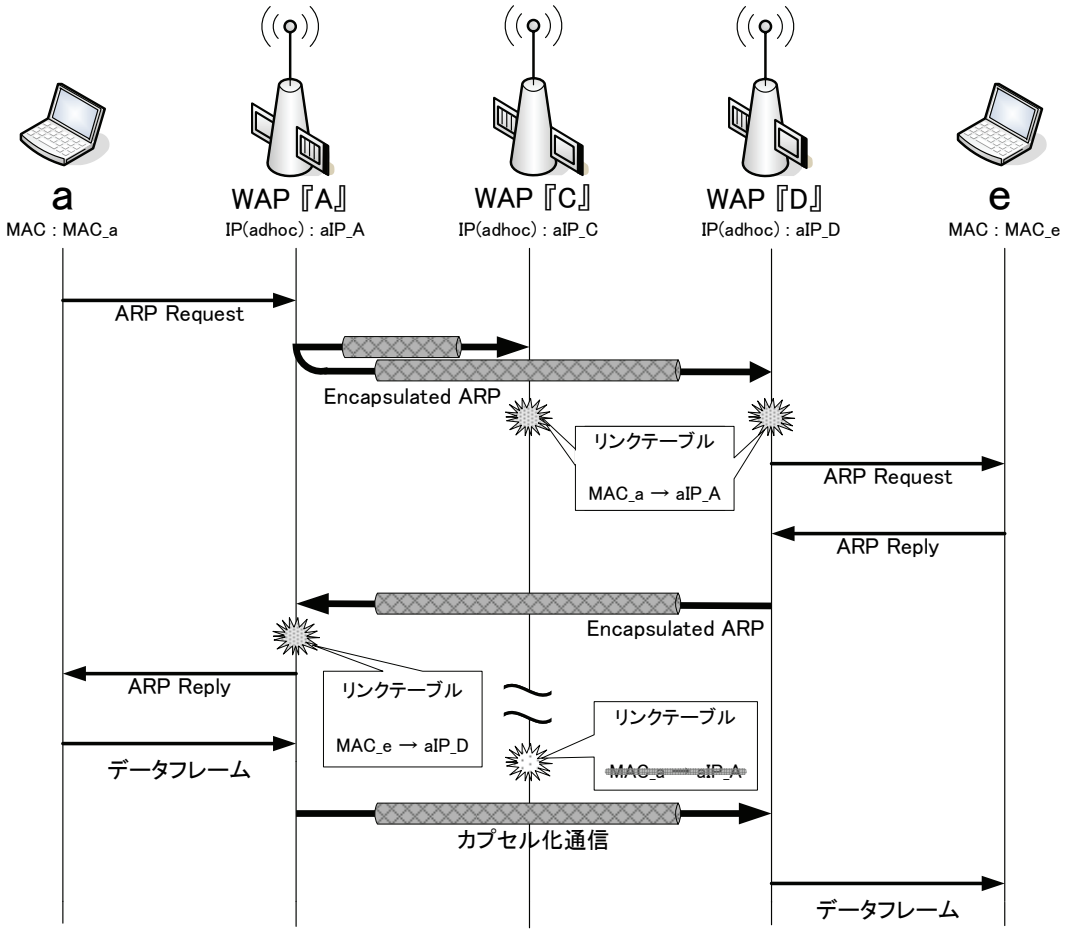


図 5　リンクテーブル作成シーケンス

dcast）

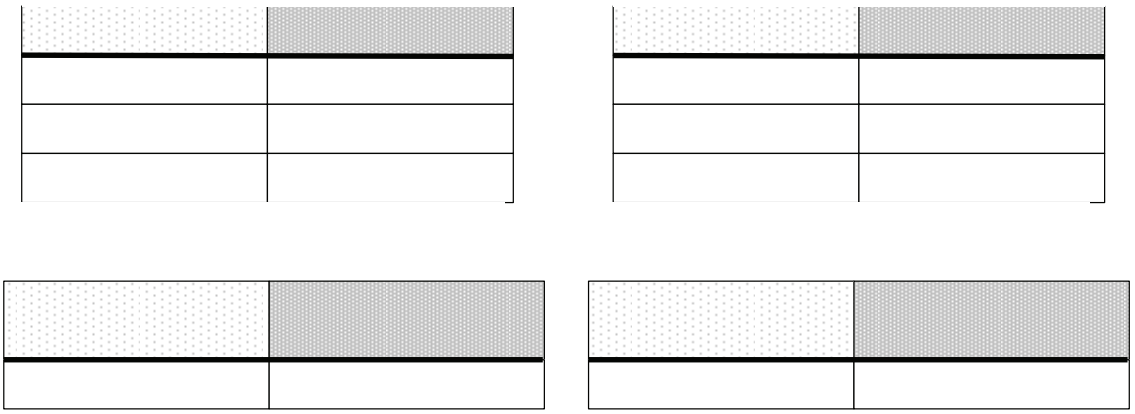


図 6　WAP の保持するテーブル内容

生成

端末 a は通信開始に先立ち，ARP 要求パケットをブロードキャストする．上記パケットを受け取った WAP『A』は，これをマルチキャスト IP アドレスでカプセル化し，他の全 WAP へフラッディングする．各 WAP はカプセル化を解除して ARP 要求パケットを配下の端末にブロードキャストする．同時に受け取った ARP 要求パケットの情報から「端末 a は WAP『A』の配下に存在する」というリンクテーブルを生成する．

配下に宛先端末 e を持つ WAP『D』は，端末からの ARP 応答パケットを受け取る．WAP『D』は先ほど生成したリンクテーブルを参照し，WAP『A』の IP アドレスで ARP 応答パケットをカプセル化して，WAP『A』に送信する．WAP『A』はこれを受け取ると，デカプセル化を行い，端末 a に送信する．同時に ARP 応答パケットの情報から「端末 e は WAP『D』の配下に存在する」というリンクテーブルを生成する．

WAP『A』と WAP『D』が保持するルーティングテーブルとリンクテーブルは図 6 の通りとなる．以後はリンクテーブルに基づいて，パケットを宛先の WAP アドレスでカプセル化/デカプセル化することにより，端末間通信が可能になる．なお，リンクテーブルは一定時間参照されなければ自動的に削除される．

第4章 WAP の構成と実装

第4.1節 WAP の構成

WAP の構成を図 7 に示す．WAP は APF（Access Point Function）と CAPF（Capsulation Function）の，２つの機能から構成される．APF はインフラストラクチャ側のインタフェースにあたり，無線パケットと Ethernet パケットの変換を行う．CAPF は Ethernet パケットのカプセル化デカプセル化を行う．CAPF により WAPL は Ethernet を完全にエミュレートする．

WAP 内のデータの流れについて説明する．APF では配下端末からの無線パケットを Ethernet パケットへ変換し，CAPF へ転送する．CAPF ではリンクテーブルを参照し，宛先 WAP の IP アドレスにより受け取った Ethernet パケットごとカプセル化を行う．カプセル化されたパケットは，データパケットとしてアドホック側インタフェースから送信される．上記パケットを受信した WAP では，CAPF によるデカプセル化が行われ，Ethernet パケットを取り出す．取り出されたパケットは，そのまま Ethernet へ送られる．APF では受け取ったパケットを無線パケットへ変換し，配下端末へと送信する．

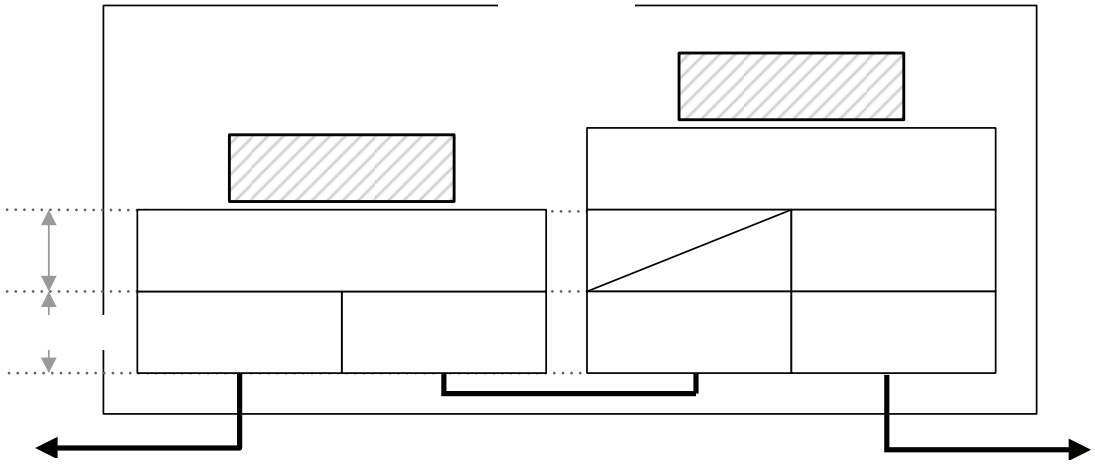


図 7　WAP のアーキテクチャ

このように，CAPF によって Ethernet を完全にエミュレートすることにより，端末は WAP を意識することなく，通常の Ethernet による環境と同様の通信を行うことができる．それ故に WAPL では，DHCP[14]，DNS，デフォルトゲートウェイなどのネットワーク技術をそのまま適用することができる．

また，端末の移動時の動作において，AP のベンダによっては AP 間で独自に情報を交換し効率的なハンドオーバーを実現している．この手順もそのまま WAPL に適応することが可能である．

WAPL では端末の IP アドレスの取得に DHCP を用いる．DHCP による IP アドレス取得時には ARP が働かないため，その時点ではリンクテーブルが生成されない．そこで，WAP では DHCP の最初のパケット（DHCPDISCOVER）を検出した際も，ARP パケットと同様にリンクテーブルを生成することとしている．なお，DNS とデフォルトゲートウェイの動作に関しては，WAP として特に特殊な処理は行わない．DNS サーバとデフォルトゲートウェイの IP アドレスは DHCP による自動設定を利用できる．

WAP はこのようなアーキテクチャであることから，IP 層には何ら手を加えておらず，アドホックルーティングプロトコルは自在に選択することができる．

第4.2節　実装

試作 WAP として，APF に市販 AP，CAPF に FreeBSD（5．4-RELEASE）を搭載した PC を用い，Ethernet ケーブルで結合することにより実現した．CAPF は FreeBSD のカーネルを改造することで実装を完了している．今回の実装では，ルーティングプロトコルとしては，Proactive 型の OLSR を採用した．

図 8 に CAPF の実装概要を示す．実線は APF 側からパケットを受信したときの処理，点線はアドホック側インタフェースからパケットを受信したときの処理を示す．APF 側から Ethernet パケットを受信すると，データリンク層の ether_input 関数により CAPF へそのまま渡される（①）．CAPF ではリンクテーブルを参照して相手 WAP の IP アドレスを判別し，IP 層へ渡す（②）．これにより Ethernet パケットは IP ヘッダでカプセル化され送信されていく．アドホック側インタフェースからパケットを受信すると，IP 層で IP ヘッダが除去され，Ethernet パケットが CAPF に渡される（A）．CAPF は Ethernet パケットをそのままデータリンク層の ether_output 関数へ渡す（B）．これにより Ethernet パケットが APF 側へと送信される．

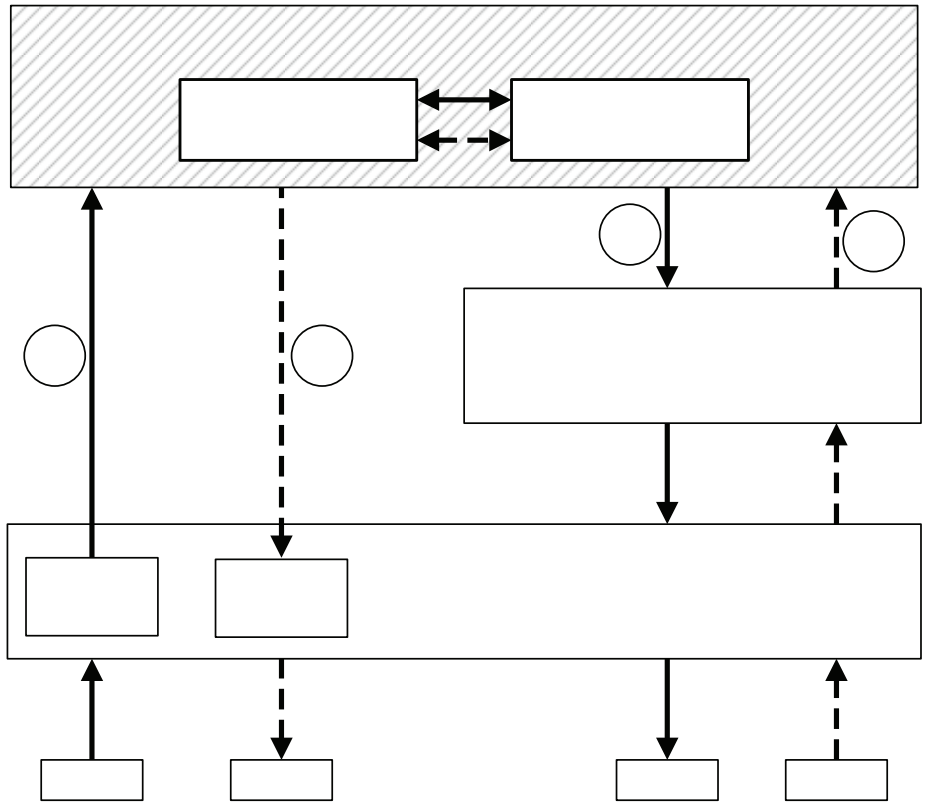


図 8　CAPF の実装概要

第5章 評価

WAPL はルーティングプロトコルを変えることにより，様々な状況に適応することが出来る．そこで本章では，ネットワークシミュレータ ns2[15]を用い，様々な条件下においてルーティングプロトコルの性能を評価する．また，WAPL と他のシステムとを比較・検討する．

第5.1節 ルーティングプロトコルの選定

MANET のルーティングプロトコルは，経路生成方法において２つの特性に分けられる．通信開始時に経路情報を生成する Reactive 型，定期的に制御パケットを出し合い，経路情報を予め生成する Proactive 型である．Reactive 型では AODV が，Proactive 型では OLSR が代表であり，ともに RFC となっている（付録参照）．ゆえに，この２つのルーティングプロトコルを比較対象として，様々な条件下でシミュレーションを実施した．なお，OLSR には UM-OLSR[16]を使用する．

(I) シミュレーション諸元

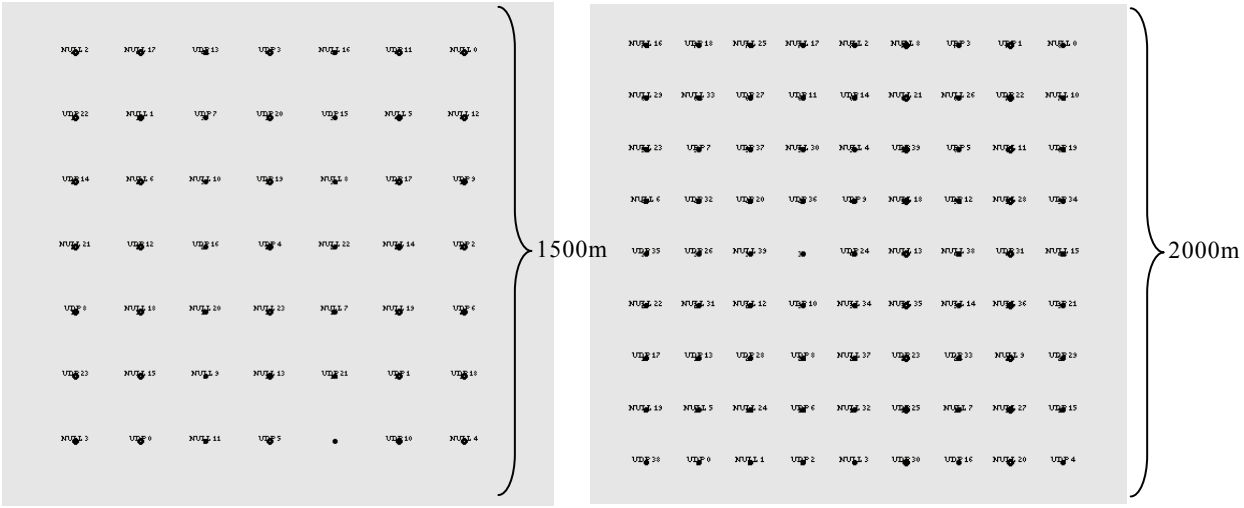
シミュレーション諸元を表 1 に示す．今回のシミュレーションでは，測定エリアを 1500m 四方，2000m 四方の 2 パターンとし，それぞれのエリア全てをカバーするように WAP を配置する．WAP の配置間隔は 200m．電波到達範囲は 250m である．WAP の必要数は，1500m 四方では 49 個，2000m 四方では 81 個となる（図 9）．

本来は，WAP の周りに端末が配置され，端末同士の通信におけるトラヒックを測定するシミュレーションモデルを想定しているが，本論文ではそこまでのシミュレーションプログラムを実現できなかった．そこで，WAP がパケットを送受信し，WAP 自身が端末をかねることで WAPL を擬似的にシミュレートした．WAPL では，端末側と WAP 側で使用するチャネルを変えることで，それぞれの通信を独立させることができる．よって，この方式でも WAPL のシミュレーションを行うに十分であると考えられる．

各 WAP はシミュレーション開始後，1 秒毎に任意に選ばれた 1 ペア（送信元と宛先のペア）が通信を開始していく．1500m 四方の場合では開始 21 秒で，2000m 四方の場合では開始 41 秒で全ての WAP が通信ペアとなる．ns2 の通信用アプリケーションとしては､送信元 WAP に CBR(Constant Bit Rate;固定間隔での送信)，宛先 WAP に NULL（パケットを受けとっては破棄するアプリケーション）を設定

する．WAP 間通信に用いる周波数チャネルはすべて同一とする．通信内容は，IP 電話を想定し，200byte 長のデータを 0.05 秒毎に UDP パケット送信する．

WAP のルーティングプロトコルに AODV，OLSR をそれぞれ適応し，各条件の下で，ルーティングプロトコルの制御トラヒック量，通信パケットのロス率を調べた．



1500m×1500m エリア　　2000m×2000m エリア

図 9　WAP の配置図

表 1　シミュレーション諸元

端末数	49，81
WAP 配置間隔	縦横 200m 間隔に配置
フィールド	1500m 四方，2000m 四方
電波到達範囲	250m
チャネルアクセス方式	CSMA/CA
無線帯域	54M
チャネルタイプ	WirelessChannel
伝搬方法	TwoRayGround
アンテナタイプ	OmniAntenna
キュータイプ	DropTail（FIFO）
最大キュー長	500
ルーティングプロトコル	AODV，OLSR
MAC	802.11
TCP・UDP	UDP

アプリケーションプロトコル	CBR，NULL
パケットサイズ	200byte
通信インターバル	0.05[秒/回]
移動モデル	固定
端末移動速度	0 [m/s]
OLSR　HELLO メッセージ送信間隔	1 秒
OLSR　TC メッセージ送信間隔	5 秒

(II)　シミュレーション結果

ネットワークの通信ペア数が変化した場合におけるシミュレーション結果を図 10-12 に示す．通信ペア数とは，エリア内における通信中の WAP の組である．本シミュレーションでは単一方向通信であるため，ペアの数だけ WAP がパケットを送信しており，同一の数の WAP がこのパケットを受信している．

図 10 は 1500m 四方エリアのシミュレーション結果を，図 11 は 2000m 四方エリアのシミュレーション結果を示す．横軸に通信ペア数，縦軸にトラヒック数を表す．トラヒック数は，ネットワークトラヒック全体と，アドホックルーティングプロトコルの制御パケットトラヒックを表す．また，図 12 では，横軸に通信ペア数，縦軸にパケットロス率を表す．パケットロス率の定義は，エリア中の全 WAP が受信したパケットの総和に対する，破棄されたパケットの比率である．OLSR は Proactive であることから経路情報を確定するまである程度時間がかかるため，シミュレーション時間は 50 秒～80 秒の 30 秒間（安定時）とした．

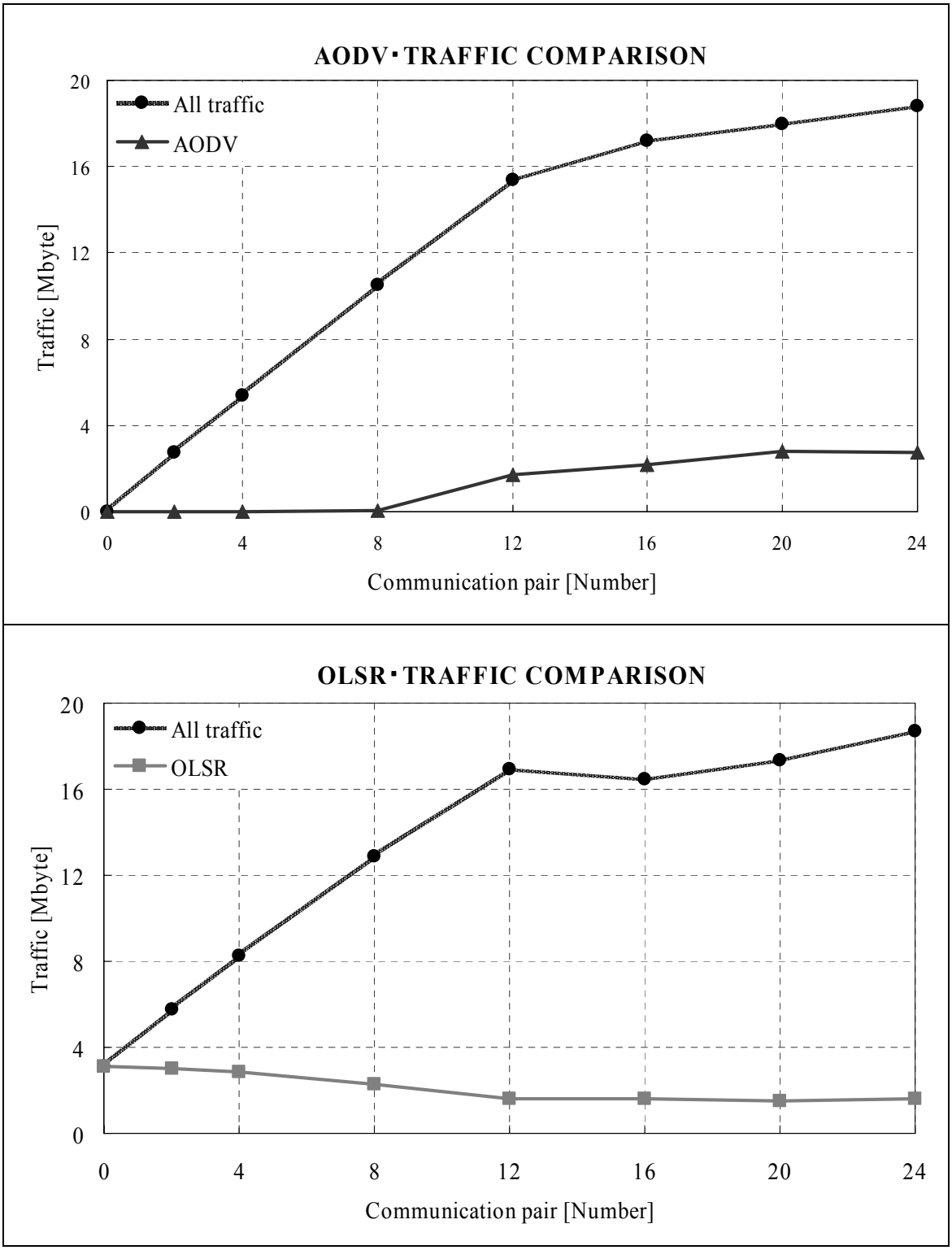


図 10　1500m 四方・通信ペア数に対するシミュレーション結果

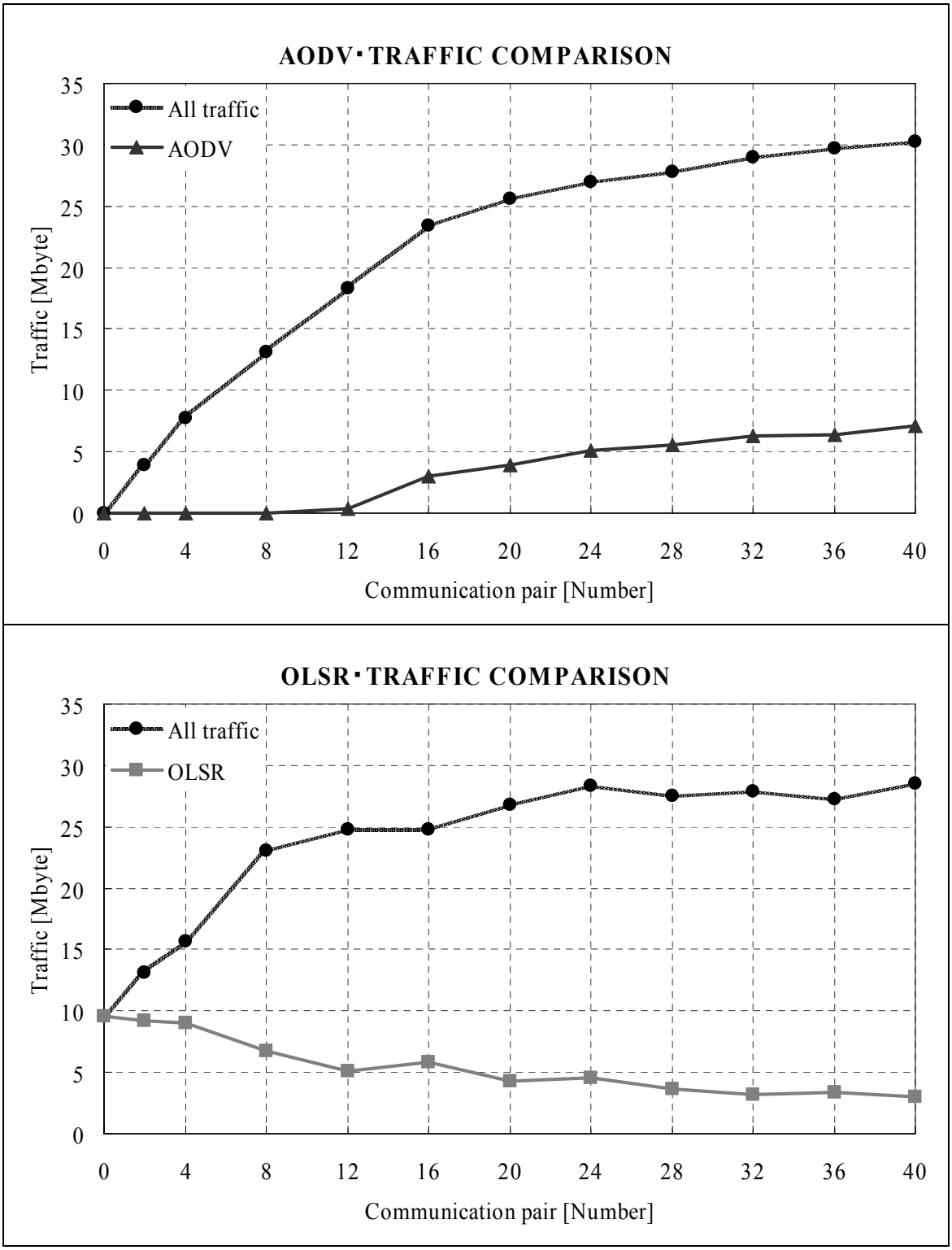


図 11　2000m 四方・通信ペア数に対するシミュレーション結果

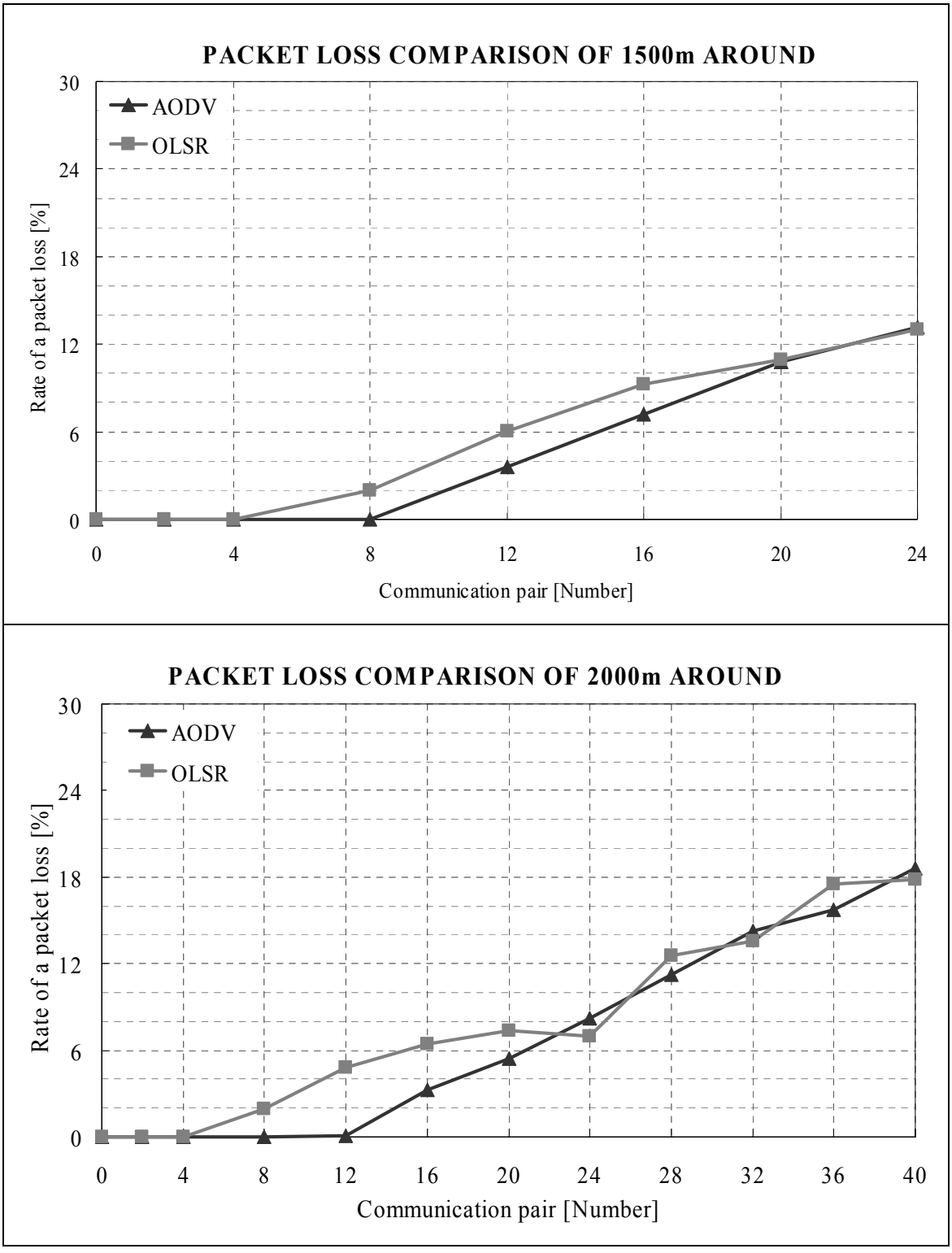


図 12　通信エリア別パケットロス率の比較

アドホックルーティングプロトコルの制御トラヒックは，AODV の場合は通信ペア数が増えるにつれて，その割合が増加する．AODV は通信開始時にのみ制御パケットをフラッディングし，経路情報を生成する．したがって，通信ペア数が 8 ペアまでは制御トラヒックが極端に増加することはない．しかし，通信ペア数が 8 ペアを超えると，AODV の制御パケットが急増する．これは，経路の輻輳により，802.11 で規定されている「RTS・CTS」方式による送信待ちのパケットが増加し，キャッシュから溜まったパケットを破棄して，経路の再構成を行う動作が働くためである．このときデータリンク層から CallBack（以後 CBK）と呼ばれる指示が上位層に出され，その度に AODV の制御パケットであるルートエラーメッセージがフラッディングされる．この処理は AODV 特有の機能である．このような機能を保持することから，AODV は端末の移動頻度が高く，頻繁に経路が切り替わるネットワークに適している．また，CBK が発生すると，パケットを破棄するので，パケットロス率が増加する．図 12 から，制御パケットのトラヒックが増加するタイミングとパケットロス率が増加するタイミングがほぼ同じであることが分かる．OLSR では，通信に先立ち，制御パケットを送信しあう．通信ペア数が増えるにつれて，アドホックルーティングプロトコルの制御トラヒックが減少する傾向にある．

(III) プロトコル評価

OLSR では，通信開始までに数回の HELLO パケット・TC パケットを送信しあい，経路情報を確定する必要がある．よって，経路確定までの間通信パケットは目的 WAP まで到達しない．本シミュレーションでは経路確定までの過渡期については評価対象から外し，通信安定時において AODV との比較を行う．

図 12 より，通信ペア数が 4 ペアを超えると，OLSR ではパケットロスが発生する．AODV では，1500m 四方では通信ペア数が 8 ペア，2000m 四方では通信ペア数が 12 ペアを超えた辺りからパケットロスが発生する．図 13 にペア数が最大時におけるパケットロスの理由別内わけを示す．CBK は CallBack によるパケットの破棄を，NRTE はルートが利用できずに生じるパケットの破棄を，IFQ はキューから溢れたことによるパケットの破棄を，TTL はパケットの生存時間が 0 となることによるパケットの破棄を，LOOP はループしたパケットの破棄を表す．AODV では CBK が，OLSR では NRTE が主なパケットロスの理由である．本シミュレーションでは，WAP は移動せず，トポロジー構成は変化しない．しかし，ルーティングプロトコルが選択した経路が重複すると，経路の輻輳が発生する．AODV では,経路の輻輳による通信待ちのパケット増加に伴い CBK が発生する．CBK が発生すると，ルートエラーメッセージを送信し，迂回経路を選択する．よって，制御パケット量は増加するが，その分パケットロス率を抑えることが出来

る．OLSR では，経路の輻輳による経路構成に変化があっても，あくまで定期送信パケットに準拠する．そのため，パケットロスの大半が NRTE 等ルートに関する理由となる．結果，両者のパケットロス率に差が生じる．AODV 特有の，トポロジー変化に強いアルゴリズムが WAPL において発揮された結果だと考察できる．

また，通信ペア数が高密度になるほど，両者のパケットロス率は増加する．通信ペア数が 20 ペアを超えると，AODV・OLSR ともに同じようなパケットロス率となる．

上記結果より，WAP を適地配備する際における，最適なアドホックルーティングプロトコルは AODV と考えられる．

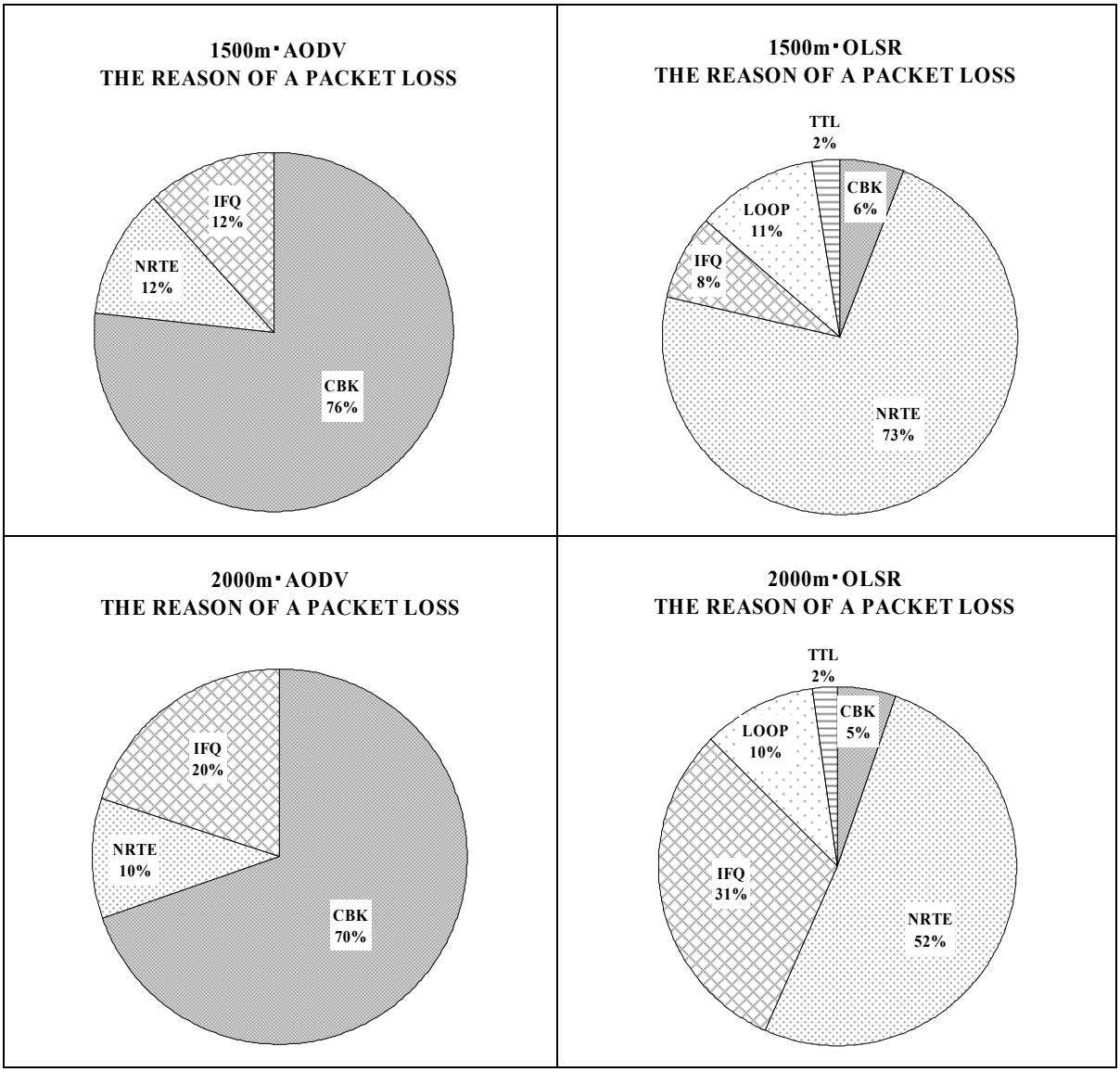


図 13　パケットロスの理由

第5.2節 従来研究との比較

表 2 に M-WLAN，WAPL の比較を示す．保持すべきテーブル量は，M-WLAN では，AP がネットワーク内の全ての端末と全ての AP の情報を保持する必要があるが，WAPL では，WAP の情報のみを MANET のルーティングプロトコルで管理し，必要最小限のリンクテーブルを生成すればよい．制御パケットによるトラヒック量は，M-WLAN では，AP 間の制御パケットに全ての情報を付加するので，端末数が多くなると，トラヒックが増大する．WAPL では，WAP の情報だけを付加するので，端末数が増加してもトラヒックに与える影響は少ない．初期遅延は，M-WLAN は予めテーブル情報を保持しているので少ない．WAPL では，リンクテーブルの生成時間分だけ初期遅延が増加するが，通信開始時だけなので実質的な影響はほとんどないと言える．端末が新たに追加される場合，M-WLAN では経路確定までの時間が定期送信されるアドホック制御パケットに依存する．よって，経路確定まで時間を要し，それまでの間は通信ができない．WAPL では AODV を採用するが，AODV は通信開始時に随時経路を確定するので多少の初期遅延は発生するものの，素早い経路構築が可能である．アドホックプロトコルは，M-WLAN では現在 OLSR を利用しているが，仮に他のルーティングプロトコルを採用しようとすると，その都度改造が必要である．WAPL では，ルーティングプロトコルが完全に独立しているため，様々なアドホックプロトコルを用途に合わせて選定することが可能である．ハンドオーバーは，M-WLAN では端末のアソシエーション情報を常に把握し，端末の移動を察知すると，アソシエーションの変更情報を送信先へ通知し，ルーティングテーブルに反映させる．そのため端末が移動時に通信パケットが流れないと，ルーティングテーブルに即座に反映できない．この場合，OLSR の制御パケットによるルーティングテーブル再構築に依存することになり再構築が完了するまで通信ができない．WAPL では，Ethernet を完全にエミュレートしているので，通常の Ethernet による環境と同等の通信を行うことができる．よって，AP のベンダが独自情報を交換し実現しているバンドオーバー技術も，そのまま適応することができる．

表 2　評価

	M-WLAN	WAPL
保持するﾃｰﾌﾞﾙ量	△	○
アドホック制御ﾊﾟｹｯﾄのﾄﾗﾋｯｸ量	△	○
通信開始までの遅延	○	△
経路確定までの遅延	△	○
アドホックプロトコルの可変性	×	○
ハンドオーバー	△	○

第6章　まとめ

メッシュネットワークを容易に構築する WAPL について提案した．WAPL では，通信開始に先立ってオンデマンドでリンクテーブルを生成する. この方式により，WAP は通信に必要最小限のテーブル情報を保持するだけでよい．WAPL はルーティングプロトコルと完全に独立した構造をとる．そのため，用途に応じてルーティングプロトコルを選定できる. また, Ethernet を完全にエミュレートするので，DHCP，DNS，デフォルトゲートウェイなどのしくみをそのまま適用できる．AP 間のハンドオーバー機能においても，Ethernet 環境で実現できる機能をそのまま WAPL に利用することができる.

次に，シミュレーションにより，WAPL に最適なアドホックルーティングプロトコルの検討を行った．適切な通信エリアを仮定し，通信ペア数を変化させたときのトラヒック量，通信パケットのロス率を調査した．シミュレーション結果から，WAP を通信エリアに適地配備する際には AODV が適していることがわかった.

今後は実機を用いて WAPL の詳細な性能を測定する．また，シミュレータに WAPL そのものを組み込み，より制度の高い評価を実施する．また，WAPL では LAN 内に自由に参加，離脱が可能であるため，フレームの盗聴や偽装等に対するセキュリティ対策が必要となる.

謝辞

本研究に関して，研究の方向や進め方など終始御熱心な御指導と御教示を賜りました，名城大学理工学部情報科学科　渡邊晃教授に心より厚く御礼申し上げます．

本論文を作成するにあたり，副査として，貴重な御意見を頂きました，名城大学理工学部情報科学科　小川明教授に心より厚く御礼申し上げます．

本論文を作成するにあたり，副査として，貴重な御意見を頂きました，名城大学理工学部情報科学科　高橋友一教授に心より厚く御礼申し上げます．

本論文を作成するにあたり，副査として，貴重な御意見を頂きました，名城大学理工学部情報科学科　宇佐美庄五講師に心より厚く御礼申し上げます．

最後に，日頃より貴重な助言を頂いた，名城大学理工学部情報科学科渡邊研究室の皆様に心より感謝致します．

参考文献

[1] K.Mase, et al., “Wireless LAM with Wireless Multihop Backbone Network”, IEEE ICWLHN 2001.pp349-358，2001

[2] 大和田 泰伯, 間瀬 憲一, “無線マルチホップ LAN の通信方式の検討とスループット評価”，電子情報通信学会　信学技報（2002）

[3] 大和田 泰伯, 間瀬 憲一, “M-WLAN における LAN エミュレータの実装と性能評価”, 電子情報通信学会総合大会, SB-9-4 (2002)

[4] 大和田 泰伯, 照井 宏康, 間瀬 憲一,“無線マルチホップ LAN のアーキテクチャにおける検討“　電子情報通信学会　信学技報 Nov. 2004

[5] http://www.tropos.com/

[6] http://www.thinktube.com/

[7] T.Clausen P.jacquet, “Optimized Link State Routing Protocol”　(OLSR) RFC3626 Oct.2003

[8] C.Perkins S.Das, “Ad hoc on-Demand Distance Vector”　(AODV) RFC3561 July 2003

[9] 市川祥平, 渡邊晃, “アクセスポイントの無線化を実現す WAPL の方式”, DICOMO2005 July.2005

[10] 小島崇広, 市川祥平, 渡邊晃, “無線アクセスポイントリンク“WAPL”の立上げ方式”,　DICOMO2005 July.2005

[11] 竹山 裕晃, 渡邊 晃, “災害時における電子メールによる安否通信方法の検討”, 情報処理学会第 67 回全国大会, March.2005

[12] 大石 泰大, 渡邊 晃, “WAPL を適用した車車間通信の実現”, 情報処理学会第 67 回全国大会, March.2005

[13] Plummer, D., "An Ethernet Address Resolution Protocol",　RFC 826, November 1982

[14] R.Droms,“ Dynamic Host Congiguration Protocol ”，RFC2131（1997）

[15] http://www.isi.edu/nsnam/ns/

[16] http://masimum.dif.um.es/um-olsr/html/

研究業績

1.学術論文

なし

2.国際会議

なし

3.口頭発表

[1] 市川祥平，渡邊晃，"アクセスポイントの無線化に関する研究"，電気関係学会東海支部連合大会，Oct. 2003.

[2] 市川祥平，渡邊晃，"アクセスポイントの無線化に関する研究"，情報通信学会　第 66 回全国大会，Mar.2004

[3] 市川祥平，渡邊晃，“アクセスポイントの無線化を実現す WAPL の方式”，情報処理学会研究報告，２００４-ＭＢＬ-３０，Ｓｅｐ．２００４．

[4] 小島崇広，市川祥平，渡邊晃，"無線アクセスポイント環境 WAPL の実現"，電気関係学会東海支部連合大会，Sep. 2004.

[5] 小島崇広，市川祥平，渡邊晃，“ WAPL における端末の IP アドレス割当て方法の検討”，情報処理学会　第 67 回全国大会，Mar.2005.

[6] 市川祥平，渡邊晃 "アクセスポイントの無線化を実現する WAPL の方式" ，DICOMO2005 シンポジウム論文集，Vol.2005，No.6，pp.225-228，Jul.2005.

[7] 小島崇広，市川祥平，渡邊晃，"無線アクセスポイントリンク“ WAPL ”の立上げ方式", DICOMO2005 シンポジウム論文集, Vol.2005, No.6, pp.221-224, Jul.2005.

[8] 山崎浩司，小島崇広，市川祥平，渡邊晃，"無線アクセスポイントリンク”WAPL”のアーキテクチャとハンドオーバーの検討", 電気関係学会東海支部連合大会，Sep. 2005.

付録

[A] アドホックルーティングプロトコルの特徴比較

MANET のルーティングプロトコルはこれまでいろいろ提案されているが，今後有望とされているのは，AODV ，DSR，OLSR，TBRPF の４種である．上記４種プロトコルの比較表を表 A に示す．

Reactive 型プロトコルはノードが常時移動することを想定しており，通信要求があった時点で経路表の生成を行う．そのため，初期遅延が大きい，経路表の収束に時間がかかるなどの課題がある．それに対し，Proactive 型は経路制御表が予め生成されているので，これらの課題は解決されているが，常時経路情報を交換しあうことから電力消費が大きいという課題がある．

次に，OLSR は定期的に経路表全体を広報するが，TBRPF はより短い周期で経路表の差分を広報する．この結果，TBRPF の方が，経路収束が早い．

表 A　ルーティングプロトコルの比較表

	AODV	DSR	OLSR	TBRPF
方式	Reactive	Reactive	Proactive	Proactive
フラッディング方法	Expanding ring serch による転送ホップ数の制限	Route Cache によるパケットの縮小	MPR 集合により効率よくフラッディング	トポロジー表を利用しパケットをルーティング
経路表作成のタイミング	通信要求時	通信要求時	定期的に情報を広報	定期的に差分情報を広報
経路表の所持方法	全ノードが持つ	パケットが持つ（送信元ノードのみ）	全ノードが持つ	全ノードが持つ
AP の密度	低い密度に有効	低い密度に有効	密度が高くてもよい	密度が高くてもよい
初期遅延	経路無しから始めるので遅い	経路無しから始めるので遅い	予め経路表があるので速い	予め経路表があるので速い
経路表の収束速度	遅い 通信要求があるまで収束しない	遅い 通信要求があるまで収束しない	中 すべての情報を周期的に送信	早い 変化した情報を利用することで、送信回数を増やす
消費電力	小	小	大	大
トラヒックへの影響	大 無造作なフラッディング	やや大 フラッディングの管理	小 MPR 集合	小 トポロジー表
ＡＰの移動	強い	強い	弱い	やや強い

[B] シミュレーションデータとグラフ

1. 通信エリアによる変化

通信エリアを変化させたときのシミュレーション結果を図 A-C に示す. 図 A は 1000m 四方エリアのシミュレーション結果を, 図 B は 3000m 四方エリアのシミュレーション結果を示す．横軸に通信ペア数，縦軸にトラヒック数を表す．トラヒック数は，ネットワークトラヒック全体と，アドホックルーティングプロトコルの制御パケットトラヒックを表す．また，図 C は通信エリア別のパケットロス率を示す．横軸に通信ペア数，縦軸にパケットロス率を表す．シミュレーション時間は 20 秒～50 秒の 30 秒間（安定時）とした．

1000m 四方エリアでは，通信パケットの量だけトラヒックが増加している．AODV において，安定時では制御パケットのトラヒックが発生しない．パケットロス率が 0%という理由からも，制御パケットが送信されなくても，通信に滞りがないと考察できる．OLSR では，一定の制御パケット量が送信される．シミュレーション結果から，1000m 四方エリアにおいて，AODV・OLSR とも問題なく通信を行うことができる．

3000m 四方エリアにおいて，AODV では 12 ペア辺りから制御パケットのトラヒックが発生している．グラフから，パケットロスの発生タイミングとトラヒックの発生タイミングが同じであることがわかる．OLSR では，8 ペアを超えた辺りから制御パケットのトラヒックが減少している．この現象については未考察である．

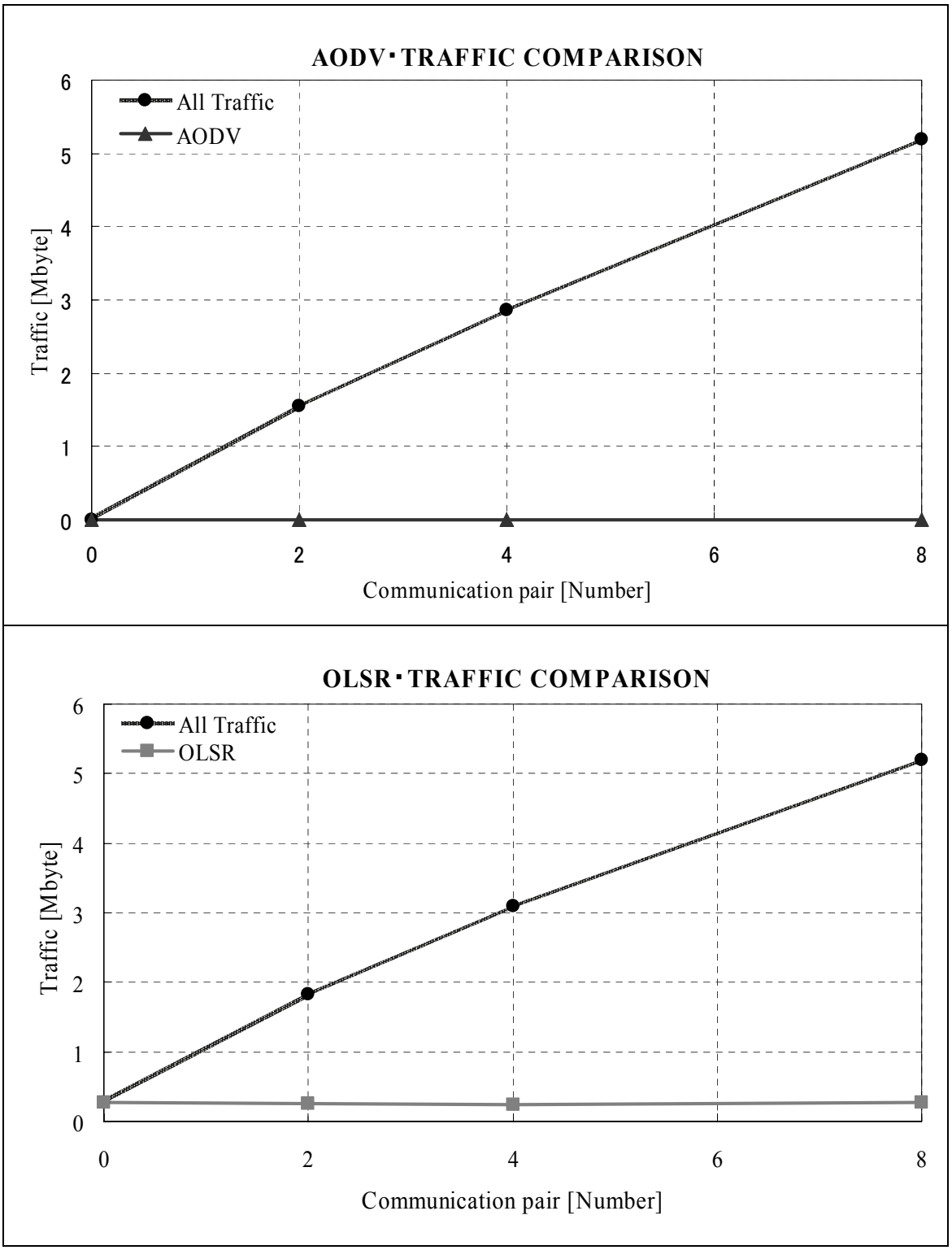


図 A　1000m 四方・通信ペア数に対するシミュレーション結果

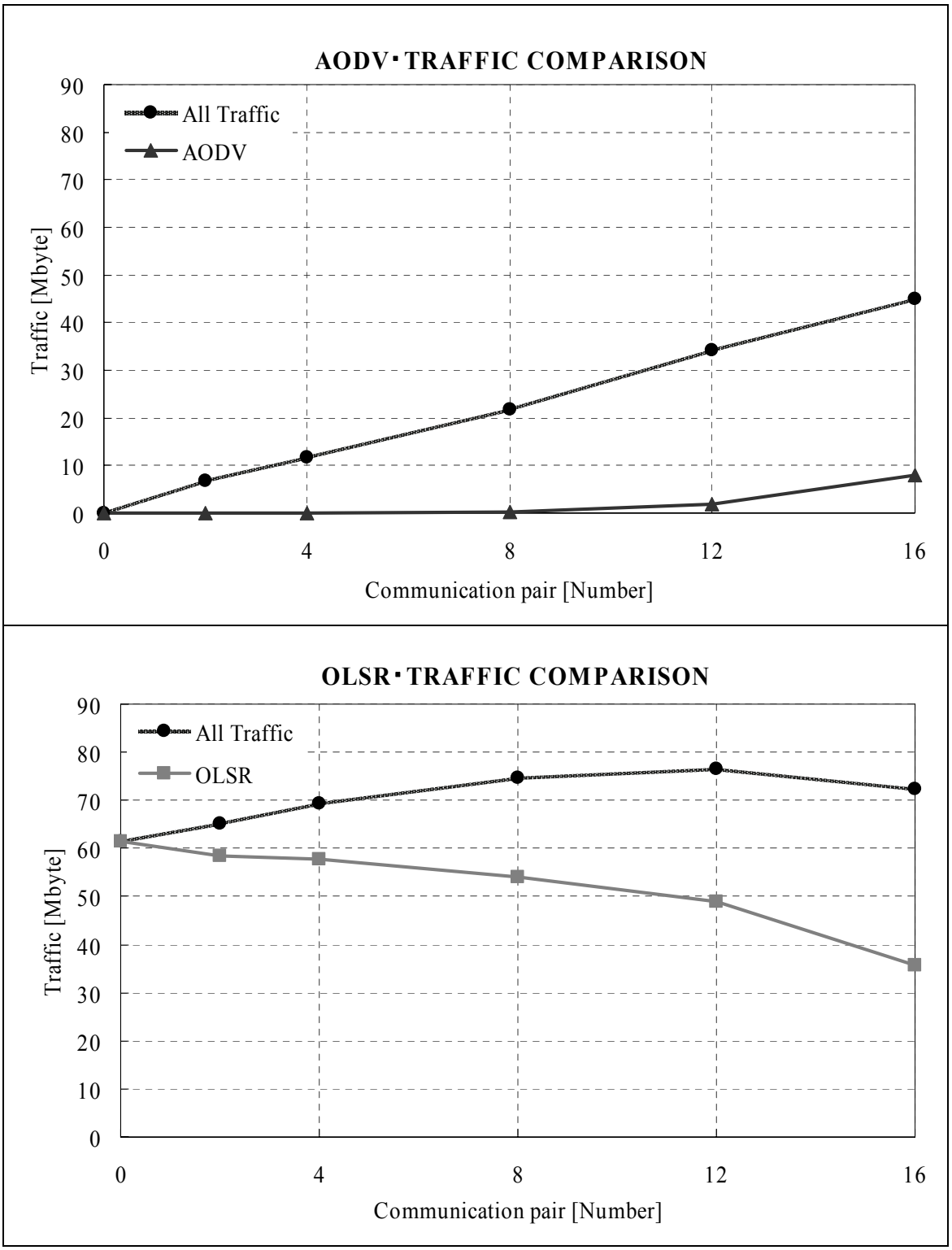


図 B　3000m 四方・通信ペア数に対するシミュレーション結果

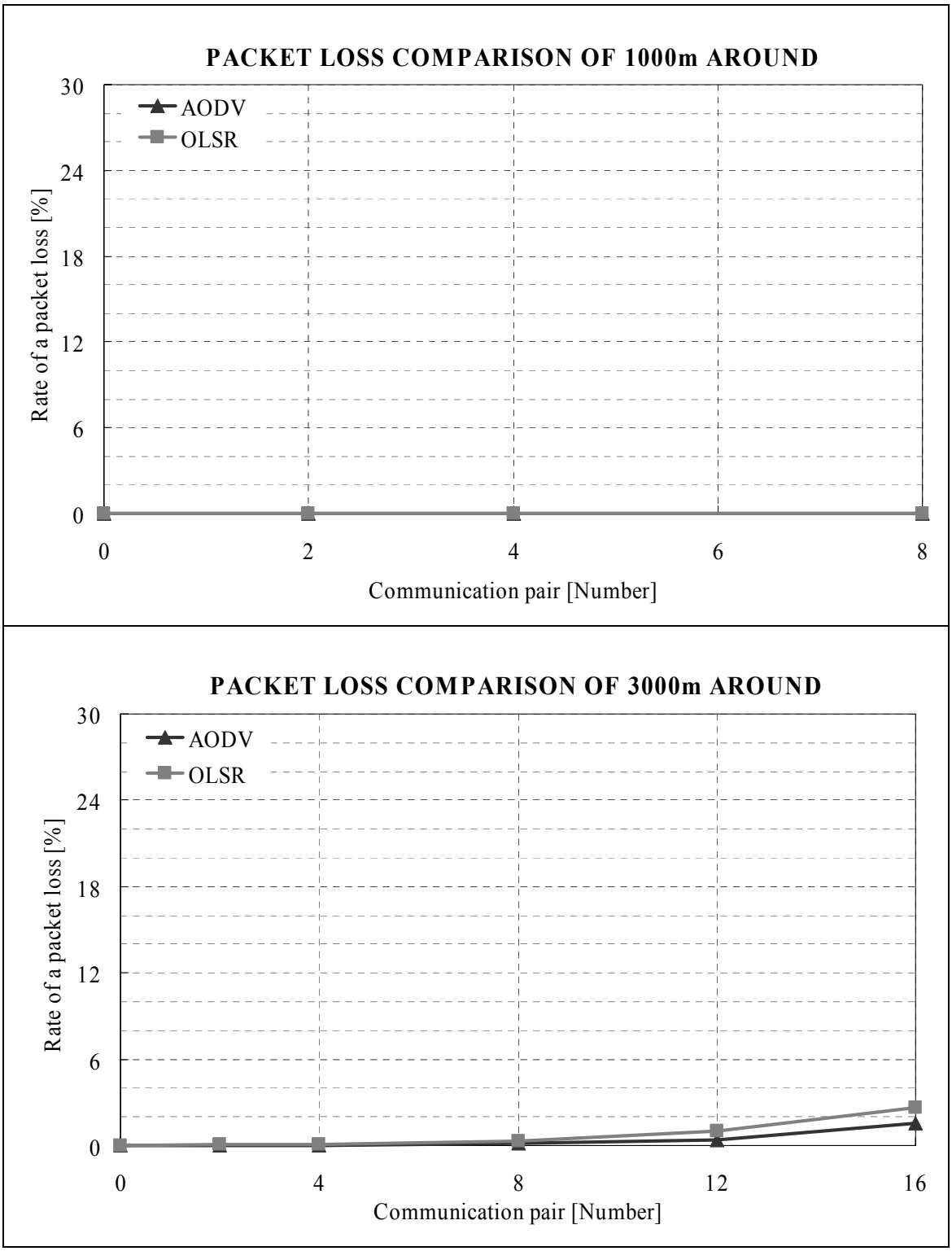


図 C　通信エリア別パケットロス率の比較

2. 帯域による変化

通信帯域を変化させたときのシミュレーション結果を図 D-F に示す．図 D は 1500m 四方エリアにおいて，通信帯域を 54M，11M としたシミュレーション結果を，図 E は 2000m 四方エリアにおいて，通信帯域を 54M，11M としたシミュレーション結果を示す．横軸に通信ペア数，縦軸にトラヒック数を表す．トラヒック数は，ネットワークトラヒック全体と，アドホックルーティングプロトコルの制御パケットトラヒックを表す．また，図 F は通信エリア別における，通信帯域を 54M，11M としたときのパケットロス率を示す．横軸に通信ペア数，縦軸にパケットロス率を表す．ルーティングプロトコルに AODV を用いる．帯域の変化におけるトラヒックの比較を求めるため，プロトコルは 1 種類でよしとした．また，シミュレーション時間は 50 秒～80 秒の 30 秒間（安定時）とした．

通信帯域が増加することで，パケットロス率が減ることがわかる．11M 時の帯域幅によるパケットロスが，54M に帯域が上がることで改善された．

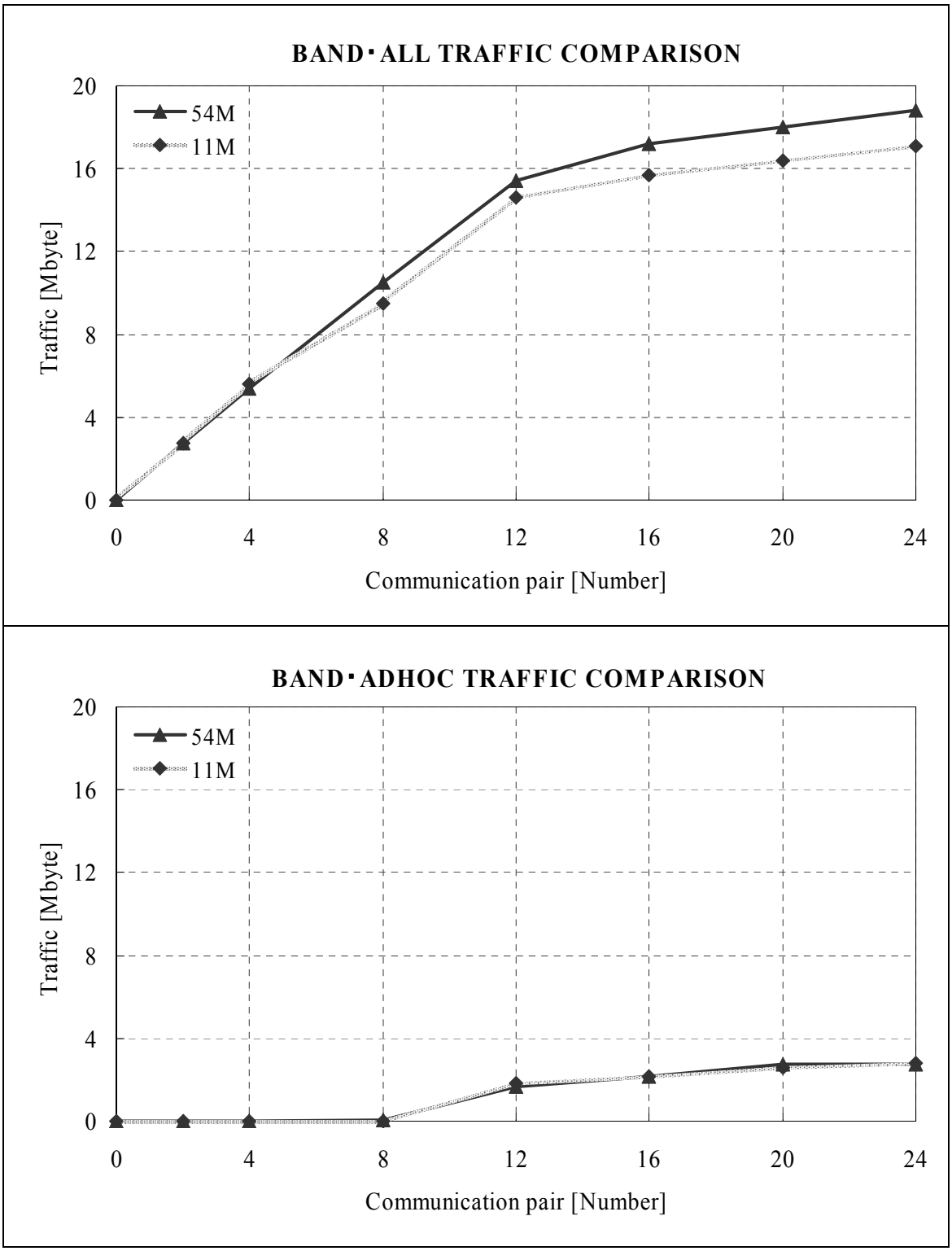


図 D　1500m 四方・通信帯域別におけるシミュレーション結果

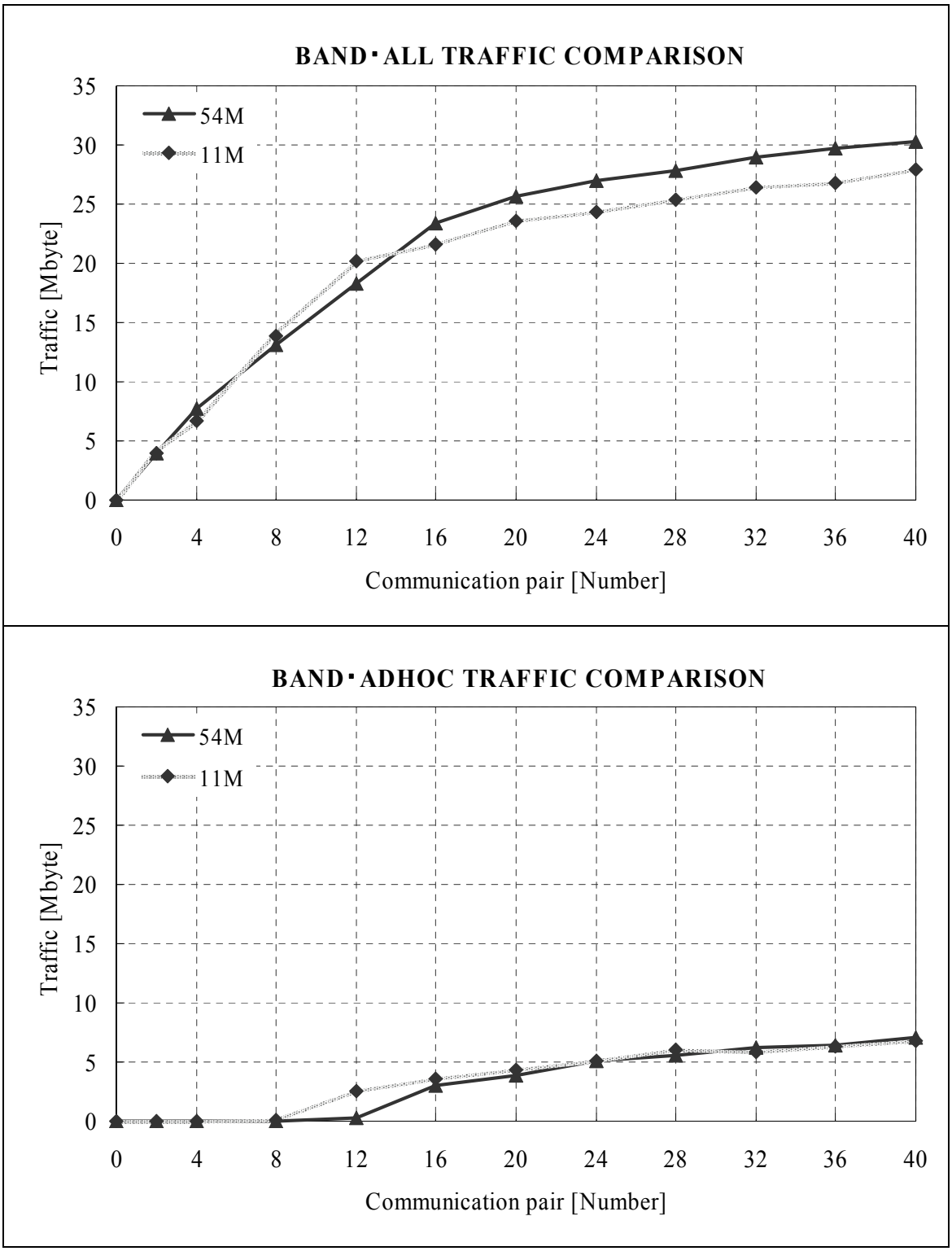


図 E　2000m 四方・通信帯域別におけるシミュレーション結果

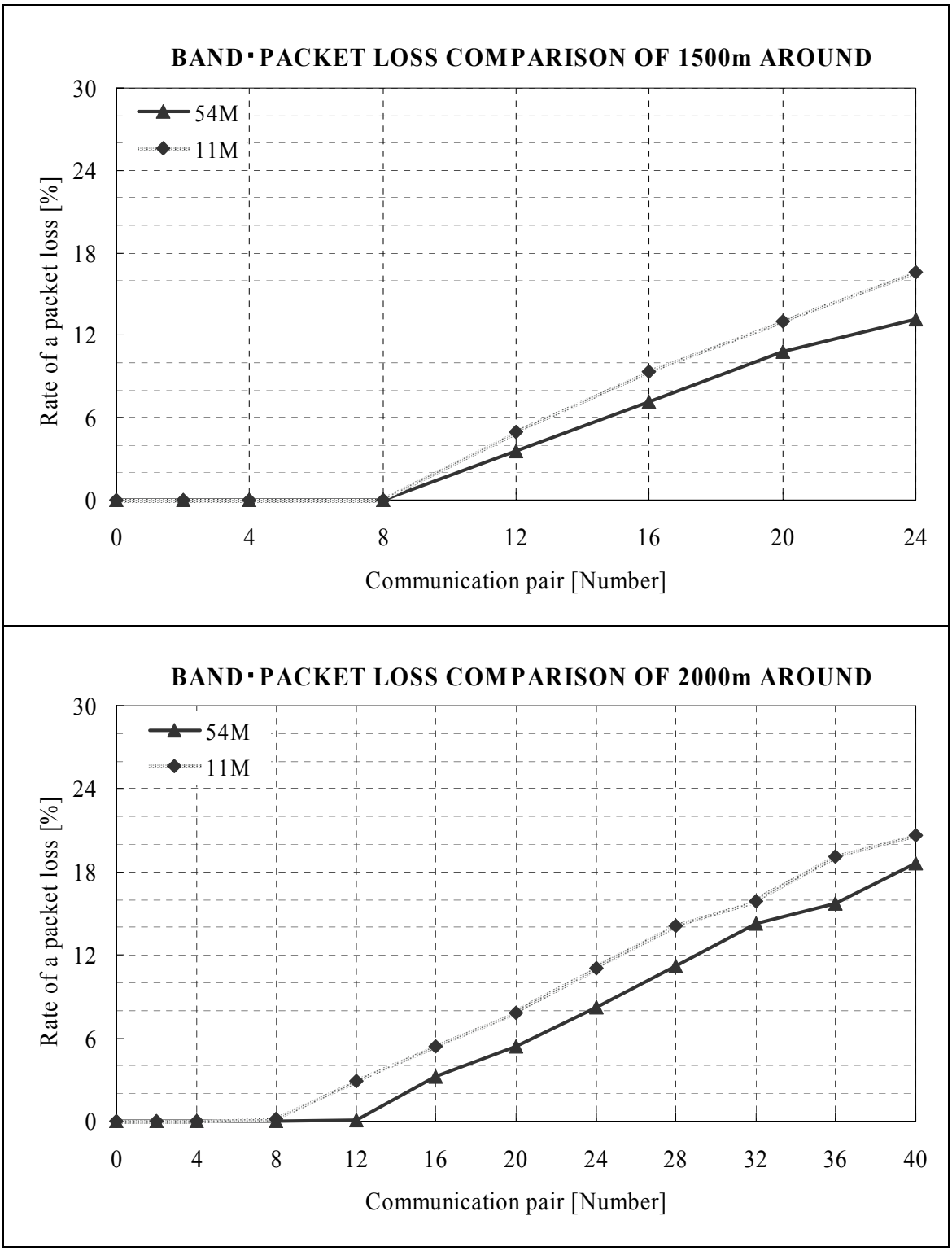


図 F　通信エリア別・通信帯域別パケットロス率の比較

3. 移動速度による変化

WAP が移動した場合において, 移動速度におけるシミュレーション結果を示す. シミュレーション諸元の変更箇所を表 B に示す. シミュレートする状況は, 1500m 四方における通信ペア数 12 ペア, 24 ペア, 2000m 四方における通信ペア 20 ペア, 40 ペアとし，20 秒から 50 秒の安定時において比較した．各状況において，各 WAP は 20 秒の時点から，10 秒毎に 1 度，方向をランダムに決定し，その方向へ直線運動を行う. 移動速度を 0m/s, 1m/s（3.6km/h：歩行速度を想定), 5m/s（18km/h：自転車速度を想定），10m/s（36km/h：自動車低速速度を想定），20m/s（72km/h：自動車高速速度を想定）とし，各速度について測定を行った．図 G に 2000m 四方エリア・通信ペア数 40 ペア・移動速度 20m/s における移動時の各 WAP の位置を示す．時間軸は 20 秒，30 秒，50 秒とした.

図 H は 1500m 四方エリアにおける 12 ペアのシミュレーション結果を，図 I は 1500m 四方エリアにおける 24 ペアのシミュレーション結果を，図 J は 12 ペアと 24 ペアにおけるパケットロス率の比較を，図 K は 2000m 四方エリアにおける 20 ペアのシミュレーション結果を，図 L は 2000m 四方エリアにおける 40 ペアのシミュレーション結果を, 図 M は 20 ペアと 40 ペアにおけるパケットロス率の比較を示す.

移動速度が上がると，トラヒック全体におけるアドホック制御パケットの量, CBR パケットのロス率が増加する．また，1500m 四方・12 ペアと 2000m 四方・20 ペアでは，パケットロス率において，OLSR が AODV を上回る．逆に，1500m 四方・24 ペアと 2000m 四方・40 ペアでは，AODV が OLSR を上回る．これは，移動することにより, ネットワーク中の WAP の密度に偏りがおこることで, OLSR の MPR 集合が効率良く選択された結果だと思われる.

表 B　シミュレーション諸元の追加

移動モデル	ランダム方向に直線移動
端末移動速度	0，1，5，10，20 [m/s]

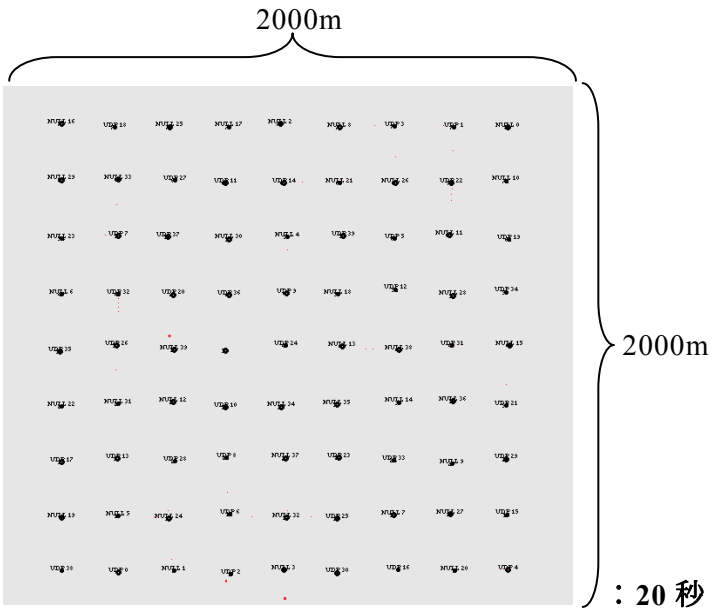


図 G　移動過程

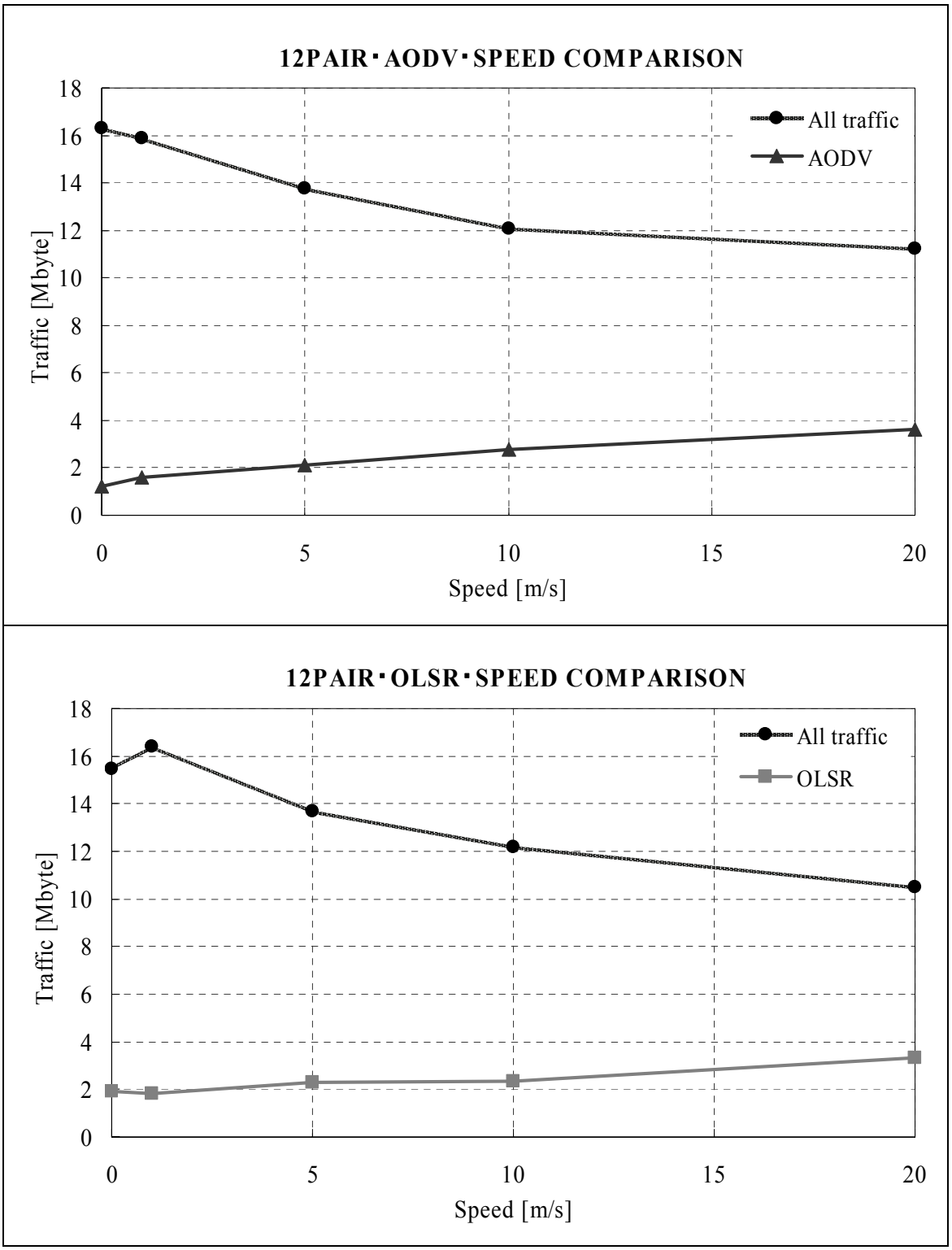


図 H　1500m 四方・12 ペア・移動速度に対するシミュレーション結果

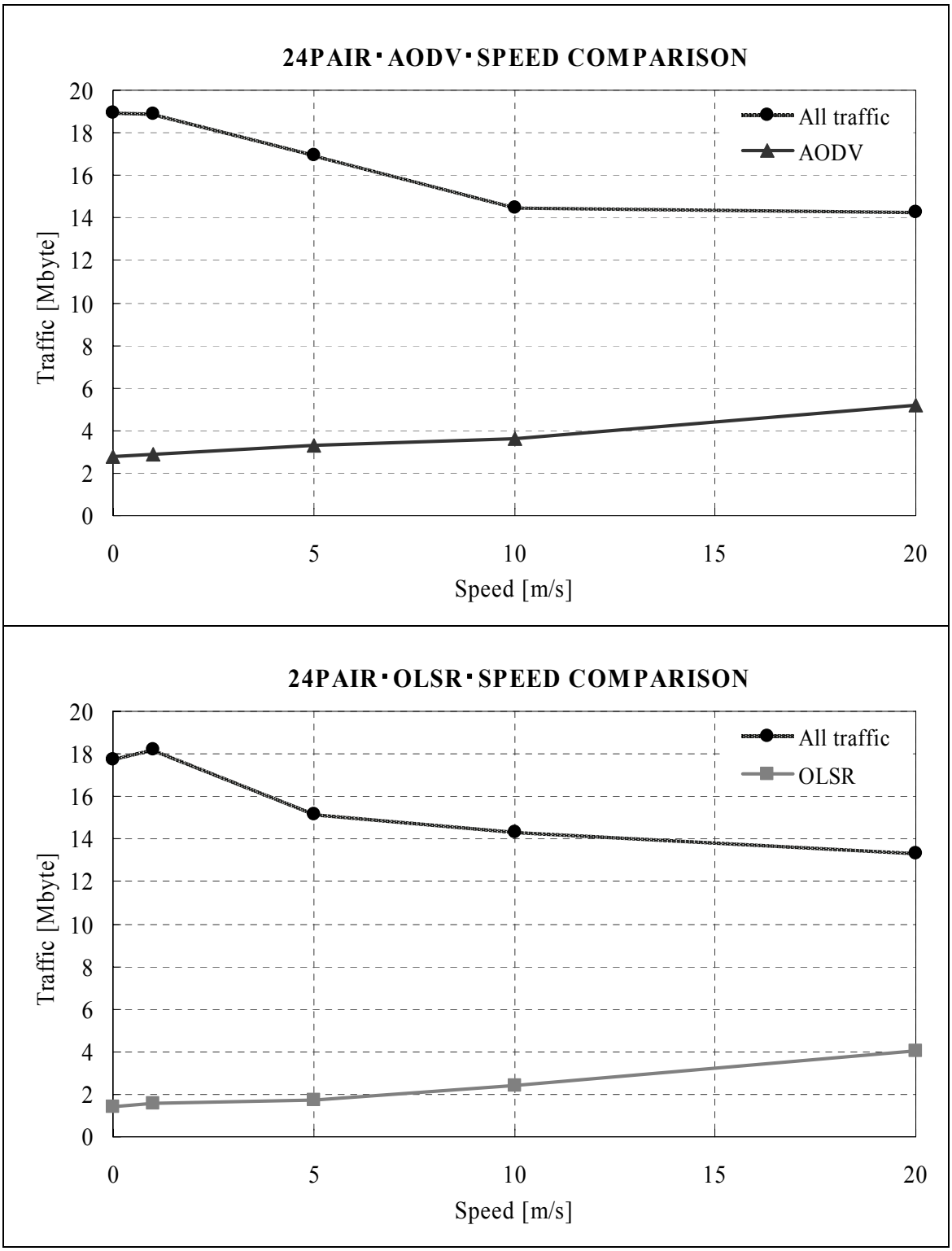


図 I　1500m 四方・24 ペア・移動速度に対するシミュレーション結果

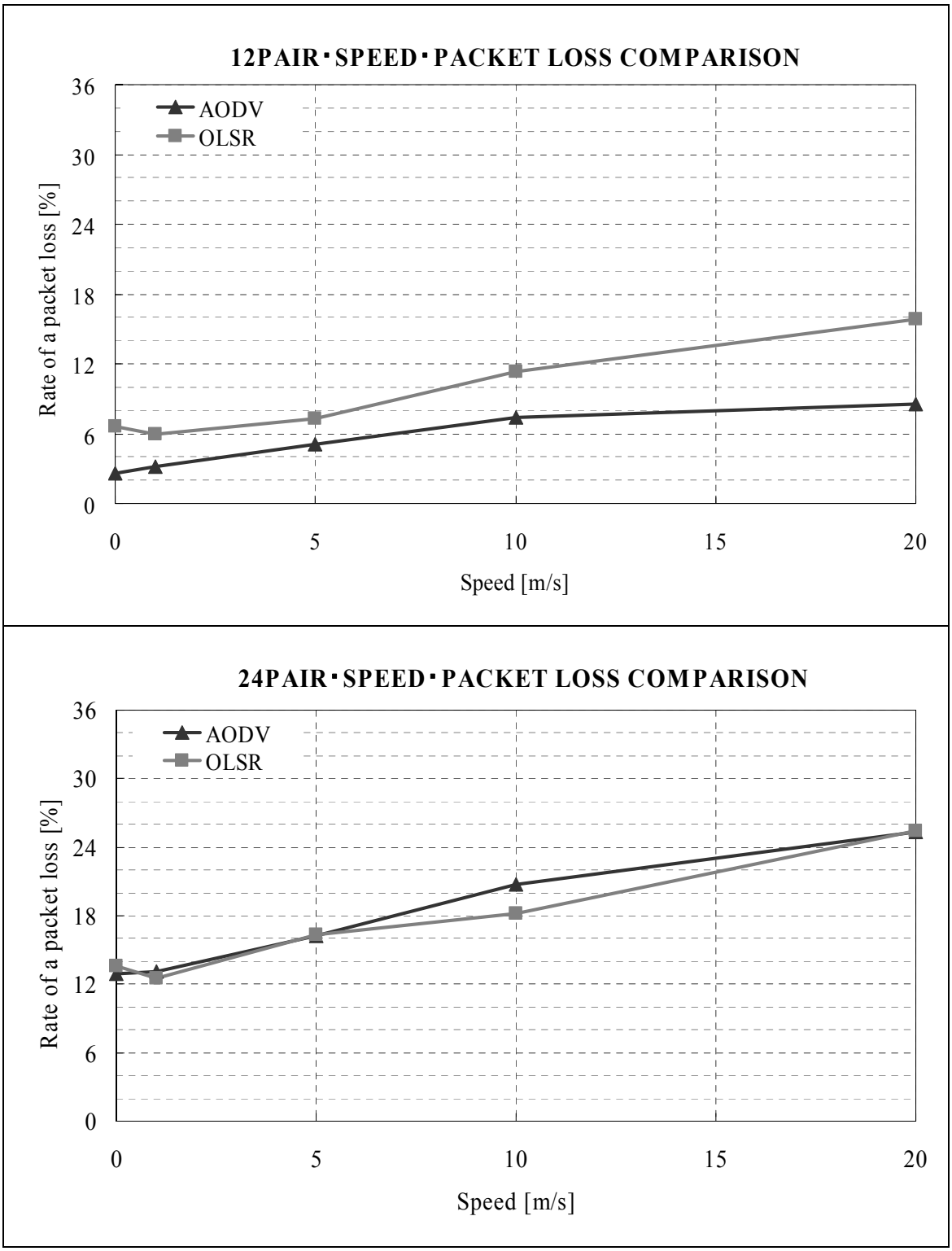


図 J　1500m 四方・通信ペア数・移動速度別パケットロス率の比較

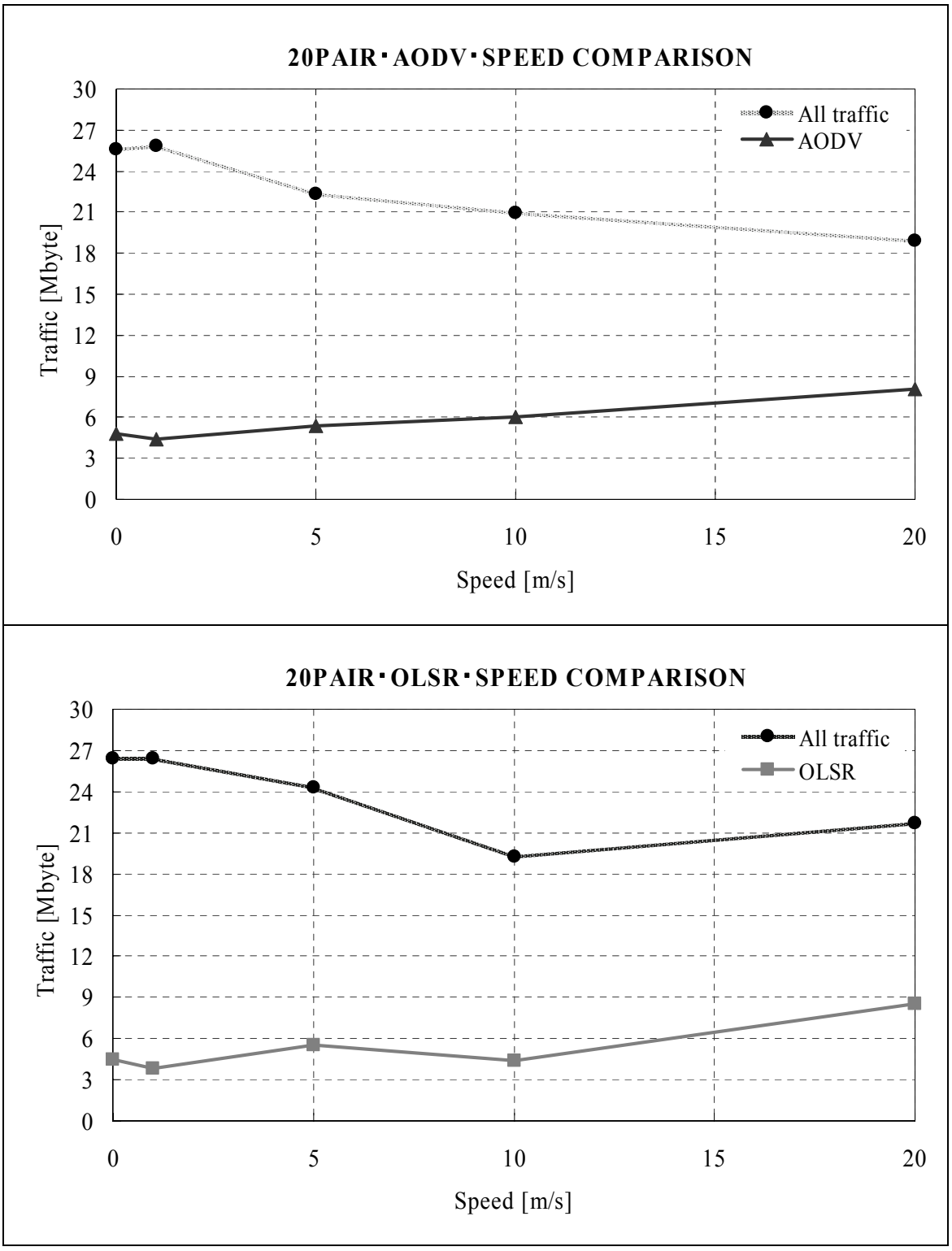


図 K　2000m 四方・20 ペア・移動速度に対するシミュレーション結果

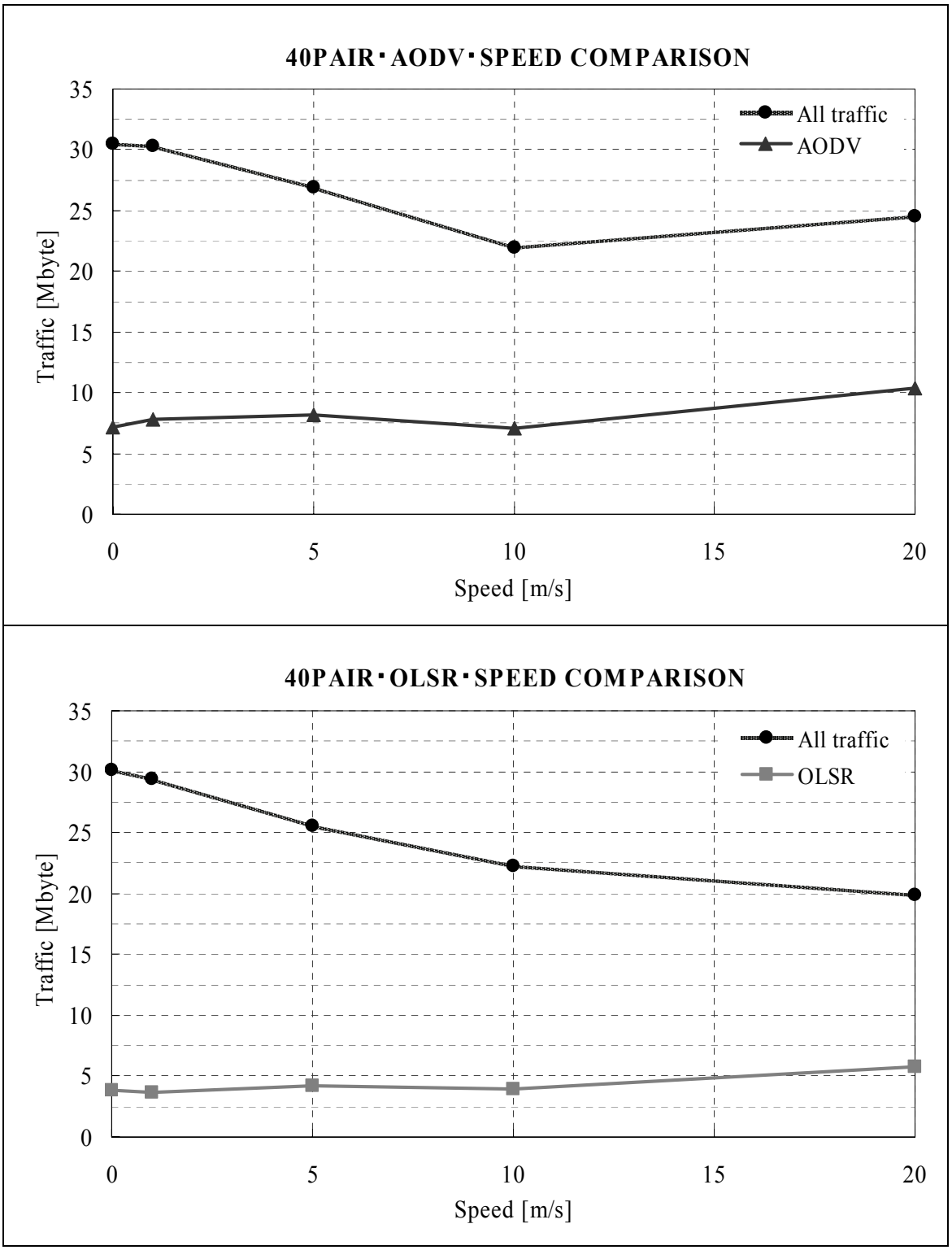


図 L　2000m 四方・40 ペア・移動速度に対するシミュレーション結果

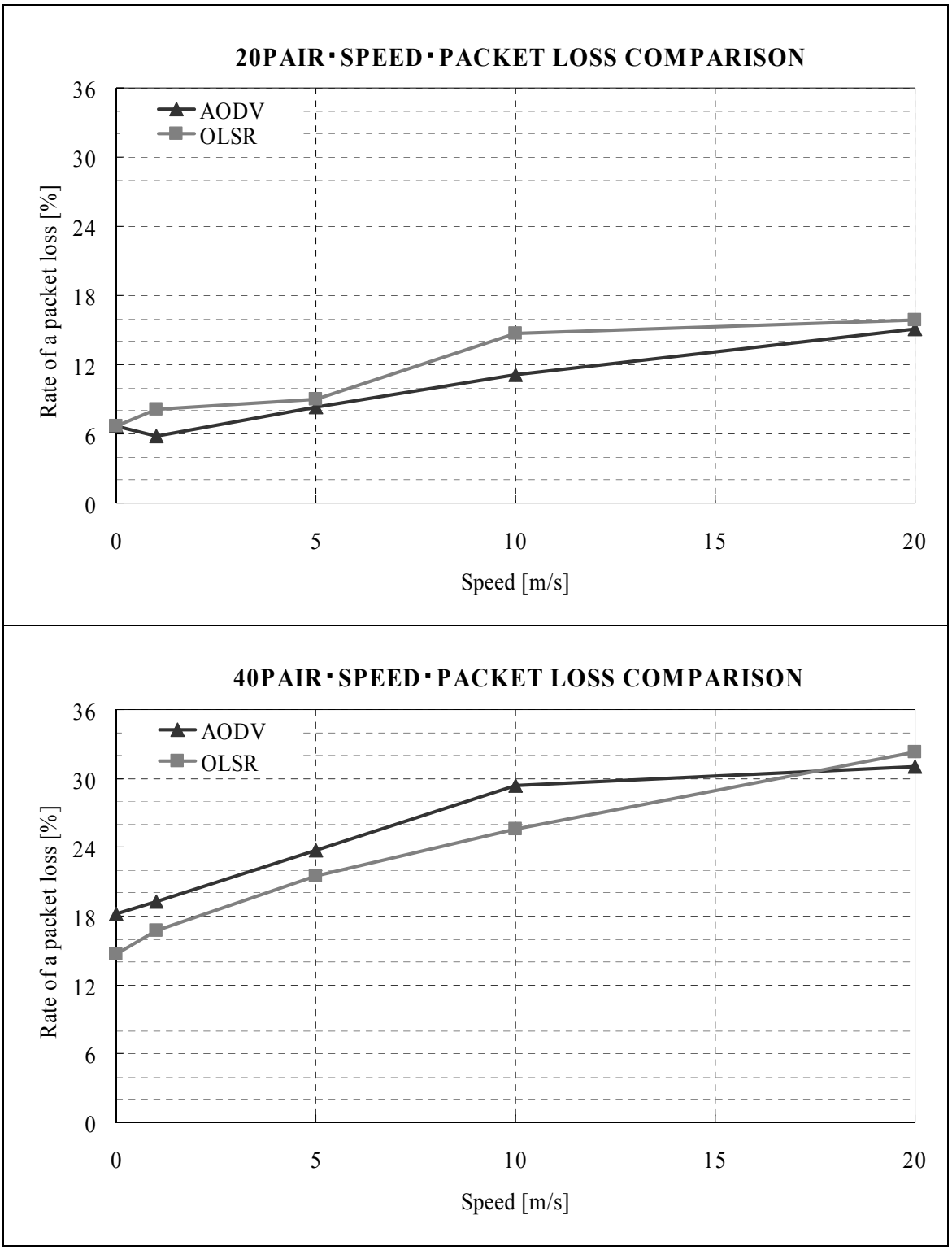


図 M　2000m 四方・通信ペア数・移動速度別パケットロス率の比較

[C] シミュレーションシナリオのソースプログラム

本論文でシミュレートしたシナリオプログラムを記載する．1500m 四方エリアにおける，通信ペア数が 12 ペア，移動速度が 5[m/s]，ルーティングプロトコルは OLSR を使用している．（図 H のシナリオ）

```
set opt(chan)           Channel/WirelessChannel      ;# channel type
set opt(prop)           Propagation/TwoRayGround     ;# radio-propagation
set opt(netif)          Phy/WirelessPhy              ;# network interface
Phy/WirelessPhy set bandwidth_ 54M
set opt(mac)            Mac/802_11                   ;# MAC type
Mac/802_11 set dataRate_ 54M
Mac/802_11 set basicRate_ 54M
set opt(ifq)            Queue/DropTail/PriQueue      ;# interface queue
set opt(ll)             LL                           ;# link layer
set opt(ant)            Antenna/OmniAntenna          ;# antenna model
set opt(ifqlen)         500                          ;# max packet in =ifq=
set  opt(nn)                49                                ;#  number  of
=mobilenodes
set opt(adhocRouting)   OLSR                         ;# routing protocol
set opt(cp)             ""                           ;# connection =pattern file
set opt(sc)             ""                           ;# node movement =file.
set opt(cpt)            10.0                      ;# Capture threshold (in dB)
set opt(cst)            1.559e-11                 ;# Carrier sense threshold
set opt(rxt)            3.652e-10                 ;# Receive threshold
set opt(lf)             1.0                       ;# System loss factor
set opt(pt)             0.2818                    ;# power of transmission (W)
set opt(rb)             2*1e6                     ;# raw bitrate
set opt(x)          1500                             ;# x coordinate of =topology
set opt(y)          1500                             ;# y coordinate of =topology
set opt(seed)       10.0                            ;# seed for random =number gen.
set opt(start)        0.1                          ;# cbr
set opt(stop)       55.0                            ;# time to stop =simulation
set opt(speed)      5.0                             ;# WAP speed
set opt(interval)       0.05                        ;# pakcet interval
set opt(size)           200                          ;# pakcet size
```

```
#
# Initialise the WirelessPhy Interface
#

Phy/WirelessPhy set CPThresh_     $opt(cpt)
Phy/WirelessPhy set CSThresh_     $opt(cst)
Phy/WirelessPhy set RXThresh_     $opt(rxt)
Phy/WirelessPhy set Pt_           $opt(pt)
Phy/WirelessPhy set L_            $opt(lf)
Phy/WirelessPhy set Rb_           $opt(rb)

#
# OLSR
#
Agent/OLSR set use_mac_       true
#Agent/OLSR set debug_          false
#Agent/OLSR set willingness 3
#Agent/OLSR set hello_ival_ 2
#Agent/OLSR set tc_ival_       5

#==================================================================
# start NS
#==================================================================

set ns_ [new Simulator]
$ns_ color 0 red
Queue set limit_    500

#=== trace & nam ==========
set tracefd    [open move_s5_olsr_54m_005_15-15_12pair.tr w]
set namtrace [open move_s5_olsr_54m_005_15-15_12pair.nam w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

#=== topo =================
```

```
set topo  [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

#=== create-god ===========
create-god $opt(nn)

$ns_ node-config -adhocRouting $opt(adhocRouting) ¥
                         -llType $opt(ll) ¥
                         -macType $opt(mac) ¥
                         -ifqType $opt(ifq) ¥
                         -ifqLen $opt(ifqlen) ¥
                         -antType $opt(ant) ¥
                         -propType $opt(prop) ¥
                         -phyType $opt(netif) ¥
                         -channelType $opt(chan) ¥
                           -topoInstance $topo ¥
                           -agentTrace ON ¥
                         -routerTrace ON ¥
                         -macTrace OFF ¥
#                          -mobileip OFF

#=== create WAP ============================
#===      WAP 0-6    ===========

    for {set i 0} {$i < 7} {incr i} {

       set node_($i) [$ns_ node]

          $node_($i) random-motion 0                    ;
          $node_($i) set X_ [expr 200+200*$i]
          $node_($i) set Y_ 200.0
          $node_($i) set Z_ 0.0

          $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
     }
```

```
#===     WAP 7-13    ===========

    for {set i 7} {$i < 14} {incr i} {

       set node_($i) [$ns_ node]

          $node_($i) random-motion 0    ;
          $node_($i) set X_ [expr 200+200*($i-7)]    ;
          $node_($i) set Y_ 400.0
          $node_($i) set Z_ 0.0

          $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
     }

#===     WAP 16-23    ===========

    for {set i 14} {$i < 21} {incr i} {

       set node_($i) [$ns_ node]

          $node_($i) random-motion 0                    ;
          $node_($i) set X_ [expr 200+200*($i-14)]
          $node_($i) set Y_ 600.0
          $node_($i) set Z_ 0.0

          $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
     }

#===     WAP 21-27    ===========

    for {set i 21} {$i < 28} {incr i} {

       set node_($i) [$ns_ node]

          $node_($i) random-motion 0                    ;
          $node_($i) set X_ [expr 200+200*($i-21)]
```

```
        $node_($i) set Y_ 800.0
        $node_($i) set Z_ 0.0

        $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
    }

#===     WAP 28-34    ===========

    for {set i 28} {$i < 35} {incr i} {

      set node_($i) [$ns_ node]

        $node_($i) random-motion 0                    ;
        $node_($i) set X_ [expr 200+200*($i-28)]
        $node_($i) set Y_ 1000.0
        $node_($i) set Z_ 0.0

        $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
    }

#===     WAP 35-41    ===========

    for {set i 35} {$i < 42} {incr i} {

      set node_($i) [$ns_ node]

        $node_($i) random-motion 0    ;
        $node_($i) set X_ [expr 200+200*($i-35)]    ;
        $node_($i) set Y_ 1200.0
        $node_($i) set Z_ 0.0

        $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
    }

#===     WAP 42-48    ===========
```

```
    for {set i 42} {$i < 49} {incr i} {

       set node_($i) [$ns_ node]

          $node_($i) random-motion 0                    ;
          $node_($i) set X_ [expr 200+200*($i-42)]
          $node_($i) set Y_ 1400.0
          $node_($i) set Z_ 0.0

          $ns_ at 0.0 "$node_($i) setdest 111.0 111.0 0.0"
      }


#==== speed ====#

    for {set i 0} {$i < 2} {incr i} {
          for {set j 0} {$j < 49} {incr j} {
                    set max 1500
                    set ransu_x [expr $max*rand()]
                    set ransu_y [expr $max*rand()]
                    $ns_ at [expr 20.0+(10.0*$i)] "$node_([expr $j]) setdest $ransu_x
$ransu_y $opt(speed)"
          }
      }

#=== create node ===========
#### traffic UDP ##########################

set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp0
$udp0 set class_ 0

$node_(1) label "UDP 0"

set udp1 [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp1
```

```
$udp1 set class_ 1

$node_(12) label "UDP 1"

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ $opt(size)
$cbr0 set interval_ $opt(interval)
$cbr0 attach-agent $udp0

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ $opt(size)
$cbr1 set interval_ $opt(interval)
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns_ attach-agent $node_(48) $null0

$node_(48) label "NULL 0"

set null1 [new Agent/Null]
$ns_ attach-agent $node_(36) $null1

$node_(36) label "NULL 1"

###############   connect   #################

$ns_ connect $udp0 $null0
$ns_ connect $udp1 $null1

#################################################
############### paket settei ###################

$ns_ at 1.0 "$cbr0 start"
$ns_ at $opt(stop) "$cbr0 stop"
$ns_ at 2.0 "$cbr1 start"
$ns_ at $opt(stop) "$cbr1 stop"
```

```
##########################################
#### traffic UDP ##########################

set udp2 [new Agent/UDP]
$ns_ attach-agent $node_(27) $udp2
$udp2 set class_ 0

$node_(27) label "UDP 2"

set udp3 [new Agent/UDP]
$ns_ attach-agent $node_(45) $udp3
$udp3 set class_ 1

$node_(45) label "UDP 3"

set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ $opt(size)
$cbr2 set interval_ $opt(interval)
$cbr2 attach-agent $udp2

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ $opt(size)
$cbr3 set interval_ $opt(interval)
$cbr3 attach-agent $udp3

set null2 [new Agent/Null]
$ns_ attach-agent $node_(42) $null2

$node_(42) label "NULL 2"

set null3 [new Agent/Null]
$ns_ attach-agent $node_(0) $null3

$node_(0) label "NULL 3"
```

```
###############    connect    #################

$ns_ connect $udp2 $null2
$ns_ connect $udp3 $null3


############### paket settei ###################

$ns_ at 3.0 "$cbr2 start"
$ns_ at $opt(stop) "$cbr2 stop"
$ns_ at 4.0 "$cbr3 start"
$ns_ at $opt(stop) "$cbr3 stop"

#########################################
#### traffic UDP ##########################

set udp4 [new Agent/UDP]
$ns_ attach-agent $node_(24) $udp4
$udp4 set class_ 0

$node_(24) label "UDP 4"

set udp5 [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp5
$udp5 set class_ 1

$node_(3) label "UDP 5"

set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ $opt(size)
$cbr4 set interval_ $opt(interval)
$cbr4 attach-agent $udp4

set cbr5 [new Application/Traffic/CBR]
$cbr5 set packetSize_ $opt(size)
$cbr5 set interval_ $opt(interval)
```

```
$cbr5 attach-agent $udp5

set null4 [new Agent/Null]
$ns_ attach-agent $node_(6) $null4

$node_(6) label "NULL 4"

set null5 [new Agent/Null]
$ns_ attach-agent $node_(40) $null5

$node_(40) label "NULL 5"

###############    connect    #################

$ns_ connect $udp4 $null4
$ns_ connect $udp5 $null5


############### paket settei ###################

$ns_ at 5.0 "$cbr4 start"
$ns_ at $opt(stop) "$cbr4 stop"
$ns_ at 6.0 "$cbr5 start"
$ns_ at $opt(stop) "$cbr5 stop"

##########################################
#### traffic UDP ##########################

set udp6 [new Agent/UDP]
$ns_ attach-agent $node_(20) $udp6
$udp6 set class_ 0

$node_(20) label "UDP 6"

set udp7 [new Agent/UDP]
$ns_ attach-agent $node_(37) $udp7
```

```
$udp7 set class_ 1

$node_(37) label "UDP 7"

set cbr6 [new Application/Traffic/CBR]
$cbr6 set packetSize_ $opt(size)
$cbr6 set interval_ $opt(interval)
$cbr6 attach-agent $udp6

set cbr7 [new Application/Traffic/CBR]
$cbr7 set packetSize_ $opt(size)
$cbr7 set interval_ $opt(interval)
$cbr7 attach-agent $udp7

set null6 [new Agent/Null]
$ns_ attach-agent $node_(29) $null6

$node_(29) label "NULL 6"

set null7 [new Agent/Null]
$ns_ attach-agent $node_(18) $null7

$node_(18) label "NULL 7"

###############    connect    #################

$ns_ connect $udp6 $null6
$ns_ connect $udp7 $null7

############### paket settei ###################

$ns_ at 7.0 "$cbr6 start"
$ns_ at $opt(stop) "$cbr6 stop"
$ns_ at 8.0 "$cbr7 start"
$ns_ at $opt(stop) "$cbr7 stop"
```

```
########################################
#### traffic UDP ########################

set udp8 [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp8
$udp8 set class_ 0

$node_(14) label "UDP 8"

set udp9 [new Agent/UDP]
$ns_ attach-agent $node_(34) $udp9
$udp9 set class_ 1

$node_(34) label "UDP 9"

set cbr8 [new Application/Traffic/CBR]
$cbr8 set packetSize_ $opt(size)
$cbr8 set interval_ $opt(interval)
$cbr8 attach-agent $udp8

set cbr9 [new Application/Traffic/CBR]
$cbr9 set packetSize_ $opt(size)
$cbr9 set interval_ $opt(interval)
$cbr9 attach-agent $udp9

set null8 [new Agent/Null]
$ns_ attach-agent $node_(32) $null8

$node_(32) label "NULL 8"

set null9 [new Agent/Null]
$ns_ attach-agent $node_(9) $null9

$node_(9) label "NULL 9"

###############  connect  #################
```

```
$ns_ connect $udp8 $null8
$ns_ connect $udp9 $null9

################ paket settei ###################

$ns_ at 9.0 "$cbr8 start"
$ns_ at $opt(stop) "$cbr8 stop"
$ns_ at 10.0 "$cbr9 start"
$ns_ at $opt(stop) "$cbr9 stop"

#########################################
#### traffic UDP ##########################

set udp10 [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp10
$udp10 set class_ 0

$node_(5) label "UDP 10"

set udp11 [new Agent/UDP]
$ns_ attach-agent $node_(47) $udp11
$udp11 set class_ 1

$node_(47) label "UDP 11"

set cbr10 [new Application/Traffic/CBR]
$cbr10 set packetSize_ $opt(size)
$cbr10 set interval_ $opt(interval)
$cbr10 attach-agent $udp10

set cbr11 [new Application/Traffic/CBR]
$cbr11 set packetSize_ $opt(size)
$cbr11 set interval_ $opt(interval)
$cbr11 attach-agent $udp11
```

```
set null10 [new Agent/Null]
$ns_ attach-agent $node_(30) $null10

$node_(30) label "NULL 10"

set null11 [new Agent/Null]
$ns_ attach-agent $node_(2) $null11

$node_(2) label "NULL 11"

###############    connect    #################

$ns_ connect $udp10 $null10
$ns_ connect $udp11 $null11

############### paket settei ##################

$ns_ at 11.0 "$cbr10 start"
$ns_ at $opt(stop) "$cbr10 stop"
$ns_ at 12.0 "$cbr11 start"
$ns_ at $opt(stop) "$cbr11 stop"

#############################################################
#=== FINISH ===================================

$ns_ at $opt(stop).001 "puts ¥"NS EXITING...¥" ; $ns_ halt"
proc stop {} {
      global ns_ tracefd namtrace
      $ns_ flush-trace
      close $tracefd
      close $namtrace
}

puts "Starting Simulation..."
$ns_ run
```