Bloom Filterを用いたモバイルアドホックネットワーク向けルーティングプロトコルの検討と評価

小佐野 智之　内田 良隆　石川 憲洋
（株）NTT ドコモ　サービス＆ソリューション開発部

抄録：本研究ではモバイルアドホックネットワークにおいて，Bloom Filter を利用したルーティング方式を提案する．Bloom Filterは，主にコンテンツ発見手法等に用いられており，コンテンツ固有のIDをハッシュ関数に通すことで得られるハッシュ値から、各コンテンツに対応したBloom Filterが生成される．本研究において、ネットワーク上に存在するノードのIDのハッシュ値をBloom Filterに格納し、そのBloom Filterをメッセージのルーティングに用いる．シミュレーションにより，リアクティブ型ルーティングプロトコルの一手法であるDSRと提案ルーティングプロトコルの性能を比較検討し，その有効性を確認した．

Routing Protocol using Bloom Filter for Mobile Ad Hoc Network

Tomoyuki Osano　Yoshitaka Uchida　Norihiro Ishikawa
Service and Solution Development Department, NTT DoCoMo, Inc.

Abstract : We propose a new routing protocol using bloom filter and for mobile Ad Hoc network. Bloom filter is mainly used as content discovery method. In this case, bloom filter is generated from the hashed value of content's ID and used for the content discovery. In the proposed protocol, bloom filter is generated from the hashed value of a node ID (e.g. MAC address), and used for the message routing towards the node. We had compared the performance of the proposed routing protocol with DSR by simulation and confirmed its feasibility

1. はじめに

近年，移動通信端末の発達と普及が進み，移動通信端末同士で動的にネットワークを構築するモバイルアドホックネットワークへの関心が高まっている．モバイルアドホックネットワークは従来での移動無線通信ネットワークで不可欠な基地局やコアネットワークといったインフラストラクチャに依存せず，移動通信端末同士が自律的に構成するネットワークである．

期待される適用例としては，センサネットワーク，災害時通信，車車間通信，パーソナルエリアネットワーク，家庭内ネットワーク，イベントにおける一時的なネットワーク，情報交換のためのネットワークがある．このような通信環境においては，携帯電話等の移動端末から，周囲に点在するデバイスを利用することや，異なるネットワークに存在する複数のデバイス同士が連携，協調して動作するサービスなど，新しい形のアプリケーションの実現が求められるようになると考えられる．

本研究では，前述モバイルアドホックネットワークを想定し，モバイルアドホックネットワークに含まれる携帯電話や小型センサなどに適したルーティングプロトコルを提案する．具体的には非データ送信期間では，ノード間でやりとりするメッセージによるネットワーク負荷を最小限にしつつ，Bloom Filterを用いることでデータ送信時のフラッディングを抑制するルーティングプロトコルを提案する．提案ルーティングプロトコルをシミュレーションにより，既存のモバイルアドホックネットワークのルーティングプロトコルと提案ルーティングプロトコルの性能を比較評価した．

以下，2章において，本研究で想定するモバイルアドホックネットワークについての課題を整理し，3章において，提案するプロトコルの設計について説明する．4章では，シミュレーションにより提案するプロトコルの評価を行う．最後に5章でまとめを行う．

2. モバイルアドホックネットワークにおけるルーティングプロトコルの課題

本研究で想定するモバイルアドホックネットワークでは，様々なセンサや携帯端末が混在するネットワークを想定している．想定するセンサとしては電源供給型のセンサや電源非供給型のセンサを含み，さらに，各センサや携帯端末のメモリや処理能力は小規模なものに限られているものとする．また，各センサや携帯端末は固有のNode IDを持つ．そして，オーバーレイされたネットワーク上にて，Node IDを用いたルーティングにより，互いにメッセージのやりとりを行う．

このようなモバイルアドホックネットワークのルーティングプロトコルとして，リアクティブ型のDSR[1]やプロアクティブ型のOLSR[2]などが提案されている．

リアクティブ型のDSRはオンデマンド型とも呼ばれ，データ送信開始時に送信相手への経路発見プロセスを起動する．このプロセスは経路が見つかるか，利用可能なすべての経路パターンを試し終えると終了する．一度，経路が発見，確立されると，宛先への通信が不可能になるか経路自体が不必要になるまで，その経路は保持される．データ送信開始の要求後に経路発見を行うため，データ送信を行わない期間においては通信パケットによるネットワークへの負荷は発生しない．一方，データ送信開始要求から実際に通信相手へデータ

送信を行うまでに時間を要し，さらに，経路が見つかるか，利用可能なすべての経路パターンを試し終えるまで，経路発見プロセスを実行するため，送信相手が発見できない場合には余分な経路探索のための処理と時間を要するといった欠点がある．

プロアクティブ型の OLSR はテーブル駆動型とも呼ばれ，IETF によって標準化されたその他のプロアクティブ型ルーティングプロトコルとしては TBRPF[3]がある．各ノードは各ノードからネットワーク上のノードへのルーティング情報を格納するテーブルを持つ．各ノードはネットワークトポロジの変化に応じて各ノードのテーブルにある経路情報を更新し，ネットワークトポロジと経路情報の整合性を確保する．定期的にノード間で経路情報の交換を行うため，データ送信開始要求から実際に送信相手へデータ送信を行うまでの遅延は発生しないが，データ送信を行わない間もノード間で通信を行うため，ネットワークの通信負荷が発生する．どの程度の頻度で経路情報の更新を行うのか，どの範囲のノードで経路情報の更新を行うのかを検討することで，通信負荷を軽減することが研究されているが，効率性や利便性とのトレードオフとなっている．

我々の想定するモバイルアドホックネットワークでは，電源非供給型のセンサを含み，さらにはセンサ，携帯端末ともにメモリや処理能力は限られている．そのため，データ送信時と非送信時のオーバーヘッドを抑制し，さらにはセンサや携帯端末の CPU リソースなどの消費を低減するプロトコルが必要となる．よって，以下の三点を我々の想定するネットワークにおけるルーティングプロトコルの要求条件とする．

- データ送信時のオーバーヘッドと遅延をリアクティブ型のルーティングプロトコルより低減できること
- データ非送信時におけるノード間の通信量をプロアクティブ型のルーティングプロトコルより低減できること
- 各ノードの持つ情報量（経路情報など）を低減できること

3. Bloom Filter を用いたルーティングプロトコル

3.1. Bloom Filter の概要

本研究では 2 章で述べた課題を解決するために，Bloom Filter[4]を用いたモバイルアドホックネットワーク向けのルーティングプロトコルの検討を行った．Bloom Filter は，キーの存在のみをコンパクトなデータ構造で高速に判定するためのアルゴリズムである．「キーが存在していないのに存在しているという判定」(以降 False Positive と呼ぶ)を確率的に許しつつデータ構造をコンパクトにしている．Bloom Filter はビット列であり，キーのひとつひとつをハッシュ関数によりビット列に変換し，それらの論理和操作を行うことで構成される．以下に，文字列検索の例をもとに Bloom Filter の使用方法を簡単に説明する．

1) M ビットのビット長を持ったビット列と N 個のハッシュ関数を用意する．また，ビット列は初期状態では全てのビットを 0 とし，ハッシュ値は 1 から M の範囲の値とする．図 1 の例では 10 ビットのビット列と 3 つのハッシュ関数を用いている．
2) 検索を行うデータを“モバイルアドホックネットワーク”とし，“モバイル”，“アドホック”，“ネットワーク”のキーに分割する．そして，それぞれのキーをハッシュ関数 Hash()にかけ，ハッシュ値に対応するビットを立てる．図 1 では，キー“モバイル”は 3 つのハッシュ関数よりハッシュ値が 1，10，6 となり表中のモバイルのビット列では 1，6，10 番目のビットが立っている．
3) それぞれのキーのビット列の論理和操作を行い，“モバイルアドホックネットワーク”の Bloom Filter を生成する．
4) 検索するキー“アドホック”を 3 つのハッシュ関数にかける．ハッシュ値よりビット列を生成し，3)で生成した Bloom Filer と比較する．検索キーのビット列で立っているビットが，3)の Bloom Filter と同位置のビットにおいて全て立っていれば，検索を行うデータ中に検索キーが含まれている可能性がある．逆にひとつでも立っていないものがあれば，検索を行うデータ中には確実に検索キーは含まれていない．

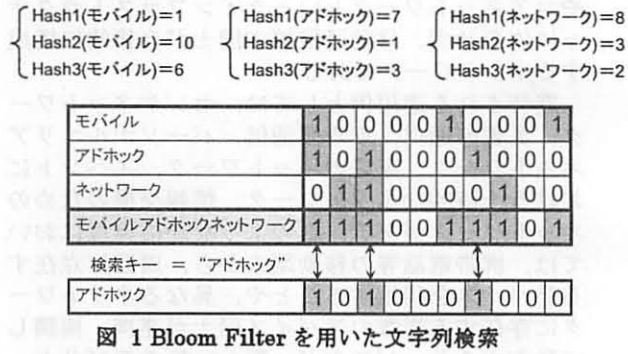

図 1 Bloom Filter を用いた文字列検索

4)で「含まれている可能性がある」としているのは，3)にて論理和操作を行っているため，いくつかの他のキーの論理和操作によって Bloom Filter が，検索キーのビット列と重複する可能性があるためである．このように，Bloom Filter は False Positive を確率的に許容する．

3.2. Bloom Filter のルーティングプロトコルへの適用

本論文で提案する Bloom Filter のルーティングプロトコルへの適用について説明する．ネットワーク上の各ノードはリンク毎に Bloom Filter を持ち，当該リンク先に存在するノードのハッシュ値

を格納しているものとする．よって，データ送信時には，目的ノードの ID（以下，NodeID）のハッシュ値によって作成された Bloom Filter と各リンクの Bloom Filter を照らし合わすことでデータを送信するリンクを決定する．具体的には，目的ノードの NodeID によって作成された Bloom Filter において立っているビットの位置が，比較するリンクが持つ Bloom Filter において立っているビットの位置に含まれる場合に，データ送信を実行する．また，各ノードは，ノード間で Bloom Filter の交換を行うことで Bloom Filter を更新する．

Bloom Filter には，Attenuated Bloom Filter[5]や，Counting Bloom Filter[6]などの機能拡張が存在するが，本研究では要求条件にて，各ノードの持つ情報量を低減することとしているため，固定のビット長かつ，値として 0 か 1 のみをとる Bloom Filter を用いる．

3.2.1. 経路情報の更新

ネットワーク上で各ノードは以下のアルゴリズムに従って Bloom Filter の更新を行う．

一般的なセンサネットワークでは，ノードが，ノード間のリンクを管理するために定期的に keep-alive メッセージを送受信する．そして，この keep-alive メッセージにより，自ノードが隣接する NodeID を管理することができる．本研究では，上記の keep-alive メッセージに自ノードの NodeID の他に，自ノードに隣接するノードの ID の情報を含めることとした．これにより，keep-alive メッセージを受信したノードは，自ノードの隣接ノードと，2 ホップ先に存在するノードの NodeID を管理することができる（図 2）．各ノードが，2 ホップ以内のネットワークトポロジを管理することで，Bloom Filter の更新や，データ送信におけるメッセージ転送の効率化を行い，メッセージ数の低減を図る．

ノードSが持つ
近傍ネットワークトポロジ情報

隣接ノード N^1	2ホップ先のノード N^2
A	D,E
B	E,F,G,H
C	G

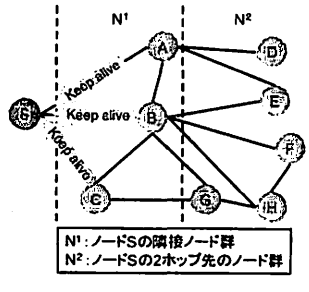

図 2 keep-alive メッセージによる 2 ホップ以内ネットワークトポロジーの管理

Bloom Filter の更新は，ネットワーク内に，新規参加ノードあるいは，離脱ノードが現れた場合に当該ノードの隣接ノードが行う．各ノードは，以下の 2 種類の Update メッセージを用いて，Bloom Filter の更新を行う．Update メッセージには，メッセージ ID を付与しており，受信ノードが，同一のメッセージ ID が付与された Update メッセージを重複して転送することを防ぐ．以下，図 3，4 を用いて各 Update メッセージの詳細について説明する．

i) Update(0)メッセージ

図 3-A は新規参加ノード B が Update(0)メッセージを用いて隣接ノードであるノード A から，当該リンクの Bloom Filter を取得する様子を示している．Update(0)メッセージは，Bloom Filter を含まず，接続先ノードに対して Bloom Filter の送信を要求する役割を担う．

① ノード B は，隣接ノードであるノード A に対して Update(0)メッセージを送信する

② ノード A は，ノード B とのリンク以外の各リンクの Bloom Filter と自分の NodeID によるビット列との論理和をとることで Bloom Filter を生成し，ノード B に送信する．

③ ノード B はノード A から受信した Bloom Filter を，受信したリンクの Bloom Filter として保持する．

ii) Update(1)メッセージ

Update(1)メッセージには 2 つの情報が含まれている．ひとつは，送信先ノードが送信元ノードとのリンクに持つための Bloom Filter であり，ふたつめは，送信元ノードが指定する送信先ノードが転送する先の NodeID の情報が含まれている．受信ノードは Bloom Filter を更新し，Update(1)メッセージを転送することで，ネットワーク全体に Bloom Filter の更新が伝播する．図 3 に，ノード A，B，C に着目した Update(1)メッセージによる Bloom Filter の更新手順を示す．

① ノード B はノード A とのリンク以外の Bloom Filter と自らの NodeID の論理和をとることで Bloom Filter を生成する．ノード B は，ノード B から 2 ホップ以内のネットワークトポロジを用い，ノード A が Update(1)メッセージを転送する先の NodeID を指定した Update(1)メッセージをノード A へ送信する．Update(1)メッセージの転送先ノードの決定方法は，(iv)にて紹介する．

② ノード A は受信した Update(1)メッセージ内の Bloom Filter を，受信したリンクの Bloom Filter として保持する．

③ ノード A は Update(1)メッセージ内に指定されたノード（ノード C）へ，Update(1)メッセージを転送する．その際，ノード A は，Update(1)メッセージ内の Bloom Filter と転送先ノード情報を，それぞれ書き換える．Bloom Filter は，送信先ノード（ノード C）とのリンク以外の Bloom Filter とノード A の NodeID による論理和へ書き換え，転送先ノード情報は，ノード A が管理する 2

ホップ以内のネットワークトポロジ情報を基に書き換える．

④ ノードCは，受信したUpdate(1)メッセージ内の Bloom Filter を受信したリンクの Bloom Filter として保持する．

iii) Update(1)メッセージ（離脱）

ネットワーク上からノードが離脱した場合，離脱ノードを検知したノードが 2 ホップ以内のネットワークトポロジを用いて Update(1)メッセージを送信する．Bloom Filter の更新手順は(ii)に従う．このとき，ノード C が受信する Update メッセージにはノード B の情報は含まれておらず，ノード C が持つ Bloom Filter からノード B の情報は削除されることが分かる．

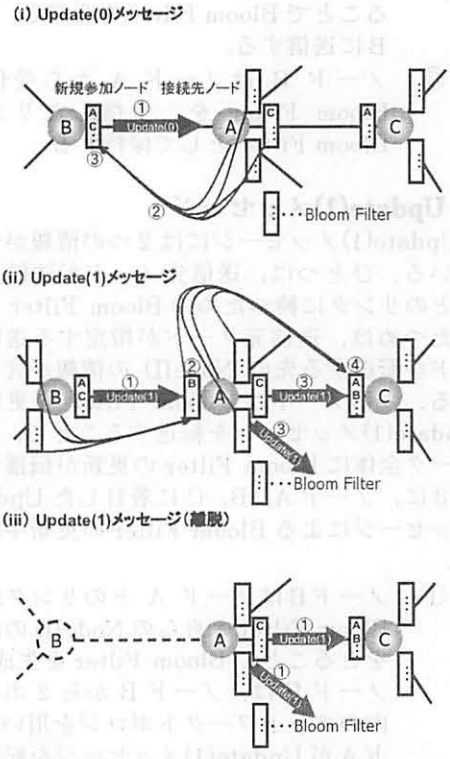

図 3 Update メッセージによる Bloom Filter の更新

iv) Update メッセージの転送方法

各ノード間での Bloom Filter の更新処理は図 3 で示した状態後も，2 ホップ先のネットワークトポロジと Update(1)メッセージを用いてネットワーク上を伝搬していく．図 4 を用い，2 ホップ先のネットワークトポロジを用いた Update(1)メッセージの転送方法の詳細について説明する．図 4 中のノード S は Update(1)メッセージの送信元ノードであり，N^1，N^2，N^3 はそれぞれ，ノード S から 1 ホップ，2 ホップ，3 ホップ先に存在するノード群を表す．

① ノードの新規参加による Update(0)メッセージの送受信が完了した後，ノード S は Update(1)メッセージを N^1 ノードへ送信する．転送先ノードの指定は，より多くの N^2 ノードを持つ N^1 ノードから順に指定する．転送先の N^2 ノードが N^1 ノード間で重複しないように指定し，N^2 ノードが，すべてカバーされた場合，残りの N^1 ノードに転送の指定を行わない．

② N^1 ノードは Update(1)メッセージを指定された N^2 ノードへ転送する．手順①と同様に転送先(N^3)の指定を N^2 ノードに対して行う．

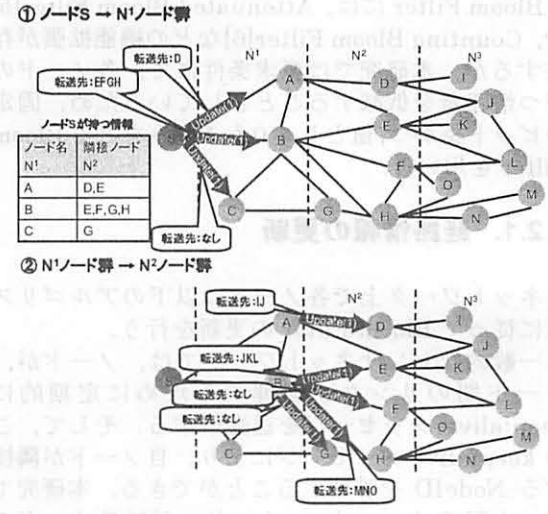

図 4 2 ホップ以内のネットワークトポロジを用いた Update メッセージの送信

上記アルゴリズムに沿うことで，各ノードはネットワーク上に存在するノードに対しての経路情報を Bloom Filter へ集約し格納していくことができる．

本研究の提案手法においては，ネットワークに変化が発生する度に，ネットワーク上に Update メッセージが流れることになる．そのため，トラヒックを最適化するため，Update メッセージ転送の操作に以下の条件を追加する．

- Update メッセージの ID が，以前に受け取った ID と一致した場合には，Bloom Filter の更新を行わず，メッセージを転送しない．
- 他ノードから受信した Bloom Filter が，現在，受信したリンクで保持している Bloom Filter と差異が無い場合，Bloom Filter の更新を行わず，メッセージを転送しない．（送信元ノードから転送先ノードが指定されている場合，指定を無視する．）

3.2.2. データ送信手順

各ノードは以下の手順で受信ノードへデータを送信する．

1. 送信ノードは受信ノードの NodeID をハッ

シュ関数にかけ，ハッシュ値をもとにビット列Aを生成する．

2. ビット列 A を各リンクにおける Bloom Filter と比較し，ビット列 A において立っているビットが，Bloom Filter と同位置で立っている場合には，
   A) 該当するすべてのリンクへデータを送信する．
   B) その他の立っているビット数が最も少ないリンクへデータを送信する
   C) その他の立っているビットが最も多いリンクへデータを送信する
   D) ランダムに(B)か(C)の条件に従ったリンクをひとつ選び，データを送信する

送信パターンは(A)～(D)の4つに分けられる。

(A) は受信ノードが存在する可能性があるリンクすべてへデータを送信するパターンであり，4つの送信パターンの中で最もメッセージ数が多くなるパターンである．(B)は False Positive の可能性が低いリンクへデータを送信することを示している．しかし，ネットワークが密になった場合には，立っているビットが増えるため，必ずしもその他のリンクの Bloom Filter より選択した BloomFilter が False Positive が低いとは言えなくなる．(C)は False Positive の可能性が大きいが，立っているビットが多いことから，そのリンクの先に接続されているノード数が多いと判断することができる．ネットワーク内の密な領域では，受信ノードも含まれている可能性が高いと判断し，データを送信することを示している．(D)は(B)もしくは(C)の条件にランダムに従いリンクをひとつ選んでデータを送信するため，(A) よりメッセージ数は少なくなる．また，(D) と (B) 両者の特徴を含むため，疎なネットワークもしくは密なネットワークに対応できる可能性がある．

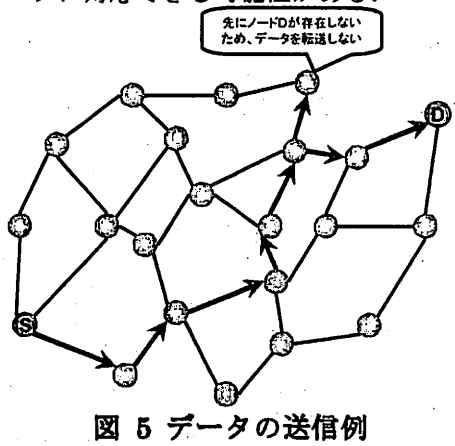

図 5 データの送信例

図5に Bloom Filter を用いたデータ送信例を示す．Bloom Filter は False Positive を確率的に許容しているために，冗長なルートが出来る可能性がある．しかし，転送先ノードにて次の転送先リンクを発見できない場合には，データの転送を停止する．データ送信パターン B，C，D では，送信経路をひとつに絞るため，誤送信の場合には送信中に転送が不可能となる可能性がある．
データ送信手順 2-B，C，D では，送信経路をひとつに絞るため，誤送信の場合には送信中に転送が不可能となる可能性がある．

そのため，再送制御として，以下の再送アルゴリズムを送信パターン(B)～(D)導入した．これらのアルゴリズムにより，データ送信時のメッセージ数は増えるが到達率は理論上 100%となる．

① 同一 ID のデータを受信した場合，送信していないリンクへメッセージを送信する．（図 6-A）
② ①を実行できない場合，データを受信したノードへ，データを戻し，A)を実行する．（図 6-B）

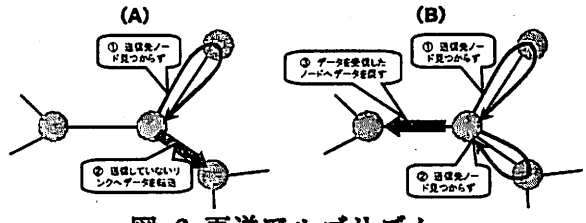

図 6 再送アルゴリズム

3.3. 他手法との比較

リアクティブ型のルーティングプロトコルと比較し，提案手法はデータ送信要求後のフラッディングによるオーバーヘッドや遅延はなく，データ送信時のネットワーク負荷を最小限に抑えることができるといえる．

また，各ノードが保持する経路情報は隣接ノード毎に持つ Bloom Filter であり，各ノードが持つ情報量をコンパクトにすることができる．さらに，本手法のアルゴリズムはプロアクティブ型のルーティングプロトコル等で用いられている Dijkstra のアルゴリズムに比べ，単純であり，各ノードでの処理量を低減できる．

4. シミュレータ実験および評価

シミュレーションにより提案する方式の有効性を評価する．初期検討としてモバイルアドホックネットワーク向けルーティングプロトコルのうち，リアクティブ型の代表として，DSR を用いて性能比較を行う．以下の点においてシミュレーション実験による比較を行う．

- DSR のルート探索におけるメッセージ数と Bloom Filter の Update メッセージ数
- データ送信時のメッセージ数．

表 1 シミュレーション実験環境

項目	DSR	Bloom Filter
最大隣接数	6	
ファーストピア数	3	
全ノード数	100	
フィールドサイズ	1.0	
電波範囲		0.15, 0.2, 0.3
Bloom Filter ビット数		600
ハッシュ関数 (SHA1) の数		3

また，シミュレーションの実験環境は表 1 のようになっている．電波範囲はフィールドサイズが縦 1.0，横 1.0 のフィールドに対して，各ノードの電波が届く範囲を割合で表している．Bloom Filter のビット数は各リンクが保持する Bloom Filter のサイズとし，[7]より False Positive の割合がおおよそ 0.1 以下になるよう設定している．

DSR のルート探索におけるメッセージ数と Bloom Filter の更新におけるメッセージ数の比較では，DSR は初期の段階で既定の全ノード数が存在し，1 ステップ毎に探索するノードと探索されるノードをランダムに選び，探索を全ノード数回（100 回）行う．Bloom Filter を用いる場合では，初期の段階ではノードが存在せず 1 ステップ毎に 1 ノードをネットワークへ追加していき，既定の全ノード数に達するまで Bloom Filter の更新を繰り返す．

データ送信時のメッセージ数の比較では，両手法とも初期段階で既定の全ノード数が存在し，1 ステップ毎に送信ノードと受信ノードをランダムに選び，データの送信を行う．Bloom Filter を用いる場合，データ送信手順は 3.2.2 の 4 通りをシミュレーションした．また，DSR を用いる場合，ノードの探索におけるメッセージ数は含まないものとする．

4.1. 考察

図 7 は全ステップが完了した時点でのメッセージの総数である．電波半径の増加に伴い，隣接するノードが増えるため，Bloom Filter 構築時に発生する Update メッセージ数は増加していく．しかし，ノードが存在しない初期段階から，ノードを追加していくことと，トラヒック量を低減する条件が存在するため，DSR によるノード探索メッセージ数より，大幅に少ないことが分かる．

図 8 の縦軸は，データを 1 回送信した時に発生するメッセージの平均数を示している．DSR を用いた場合では，平均して 5〜7 ホップで受信ノードにデータが送信されているが，Bloom Filter を用いた場合には DSR の 2 倍以上の 20 メッセージ発生している．DSR と Bloom Filter を用いた場合のメッセージ数の差分はデータ転送先に受信ノードが存在しないにも関わらず，データを転送した（False Positive）メッセージの総数とみることができ，実運用時には，Bloom Filter のビットサイズ，ハッシュ関数の数などで，False Positive を最適化する必要がある．

また，Bloom Filter を用いたデータ送信では，電波半径が大きくなるに伴いメッセージ数が減少する傾向があり，さらに，データ送信パターン C よりデータ送信パターン B のメッセージ数が大きいことが分かる．基本的に電波半径が大きくなることで，各ノードの隣接数が増え，あるノードから 2 ホップ以内に存在するノード数が増加する．よって，2 ホップ以内のネットワークトポロジを用いた Update メッセージ転送により最適な Bloom Filter が構築され，データ送信時のメッセージ数の低減に繋がっていると考えられる．また，送信パターン B が送信パターン C よりも，データ送信時のメッセージ数が多い理由として，ファーストピア数 3，最大隣接数 6 という密なネットワークでは，送信パターン(B)による効果は薄く，受信ノードがネットワークの密な領域に存在することが多いことを示している．

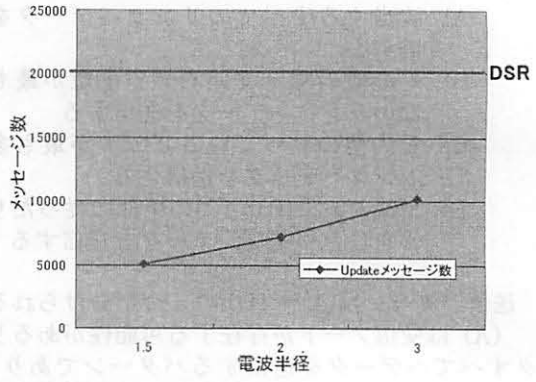

図 7 DSR のルート探索と Bloom Filter の更新におけるメッセージ数

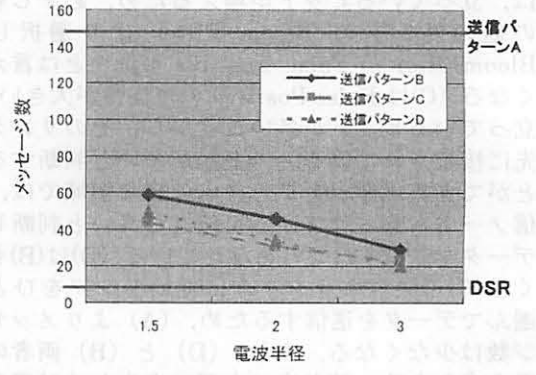

図 8 データ送信時におけるメッセージ数

5. むすび

本稿では，Bloom Filter を用いたモバイルアドホックネットワーク向けルーティングプロトコルと DSR との性能比較をシミュレーションにより行った．今後はアルゴリズムの改善を図るとともに，プロアクティブ型の OLSR などとも比較を行っていく予定である．

参考文献

1) David B Johnson, David A. Maltz, and Yim-Chun Hum, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," IETF Internet Draft. Apr. 2003.
2) T. clausen, P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Oct. 2003.
3) R. Ogier, F. Templin, M. Lewis, "Topology Dissemination Based on Reverse-Path For warding (TBRPF)," RFC3684, Feb. 2004.
4) B. Bloom, "Space/Time Tradeoffs in Hash coding with allowable errors," CACM, 13(7):422-426, 1970.
5) Sean C. Rhea, John Kubiatowicz, "Probabilistic Location and Routing," INFOCOM 2002.
6) Michel Mitzenmancher, "Compressed Bloom Filters," IEEE/ACM, Transactions on Networking, vol.10, no.5, Oct. 2002.
7) Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," IEEE/ACM Transaction on Networking, vol. 8, no 3, Jun. 2000.