MANET におけるブロードキャストに対する 不正確な位置情報の活用法とその効果

林 峻正 * 首藤裕一 * 増澤利光 *

内容梗概

動的アドホックネットワーク (MANET) において，不正確な大域情報を用いてブロードキャスト問題を解く分散アルゴリズムをいくつか提案する．さらに，平面上を移動するノードから構成される MANET におけるシミュレーションによって，それぞれの提案アルゴリズムの性能評価を行い，不正確な大域情報の有用性を示す．

1 はじめに

分散システムは，計算機（ノード）と，それらを繋ぐリンクから構成される．各ノードは，リンクを介した通信を行い，協調して動作することで，ブロードキャストや全域木構成などの，分散システム上の問題を解決する．分散システム上の問題を解決するアルゴリズムのことを分散アルゴリズムと呼ぶ．これまでに，さまざまな問題に対する効率的な分散アルゴリズムが提案されている [1, 2, 3]．分散システムの例として，インターネットや，携帯電話などの移動端末をノードとするモバイルアドホックネットワーク（MANET）などが挙げられる．

分散システムにおいて，あるノードが遠方の他のノードへメッセージを送りたいときを考える．ノードが局所的な情報しか持っていない場合は，各ノードが全ての隣接ノードへメッセージを送信する，フラディングを用いることが多い．しかし，メッセージを送信したいノードが一つであるにも関わらず，ネットワーク全体でメッセージをフラディングしてしまうのは無駄が多く，良い手法とは言い難い．

一方，各ノードが全域木や目的ノードの位置に関する大域情報を持っている場合は，フラディングを行う必要はなく，その大域情報を活用してより少ないメッセージ数で，目的ノードにメッセージを伝えることができる．しかし，移動端末から構成される動的アドホックネットワークなどでは，端末が移動するために，獲得した大域情報は時間とともに不正確になる．

このことから，不正確な大域情報を用いて大域的な問題を解く分散アルゴリズムを設計することは重要である．どのような大域情報がどの程度不正確であっても，大域問題を解く分散アルゴリズムの効率化に有用であるかを解明することは，理論的にも，また，実用の観点からも興味深い．これらの特性が解明できれば，定期的に大域情報を更新することで，MANET において効率的な分散アルゴリズムを設計できる．しかし，このようなことを目的とする研究は，分散アルゴリズムの分野ではほとんど行われていないのが実情である．

動的アドホックネットワークにおけるブロードキャストやルーティングについての関連研究はいくつかある [4, 5, 6, 7, 8, 9]．しかし，これらはどのような大域情報がどの程度不正確であっても分散アルゴリズムの効率化に有用であるかについてはあまり注目していない．

そこで，本稿では，平面上の移動ノードから構成される動的アドホックネットワークにおけるブロードキャストを対象に，ノード位置に関する不正確な情報（以前のノード位置に関する情報）の有用性について考察する．具体的には，不正確な情報を用いる分散ブロードキャストアルゴリズムをいくつか提案する．さらに，シミュレーションによって，動的モバイルアドホックネットワークにおける，これら分散アルゴリズムの完全性（どれだけのノードにメッセージを配信できたか）と通信量（受信した総メッセージ数）を評価し，不正確な大域情報の有用性を示す．

*大阪大学大学院情報科学研究科

2　諸定義

2.1　ネットワークモデル

時間とともにトポロジが変化する動的ネットワークは，離散時刻を導入し，各時刻でのトポロジを表す有向グラフの系列で表される．時刻 $t \in T = \{0, 1, 2, \ldots\}$ におけるネットワーク G を $G_t = (V, E_t)$ と表す．ここで V はノードの集合であるが，V は時刻で変化することはなく，$|V| = n$ とする．また，E_t は，時刻 t における有向辺の集合である．ノード u, v に対し，時刻 t に有向辺 (u, v) が存在する（つまり，$(u, v) \in E_t$ である）とき，ノード u から v にメッセージを送信できる．

今回扱う動的ネットワークは，平面上をノードが移動する動的アドホックネットワークである．関数 $\mathrm{vel}_t : V \to \mathbb{R}^2$ を，時刻 t におけるノード $u \in V$ の速度ベクトルを得る関数とし，関数 $\mathrm{pos}_t : V \to \mathbb{R}^2$ を，時刻 t におけるノード $u \in V$ の位置を得る関数とする．また，関数 $\mathrm{dist}_t : V^2 \to \mathbb{R}_{\geq 0}$ を，時刻 t における 2 つのノード $u, v \in V$ の間のユークリッド距離を得る関数とする．

ノードは通信半径を持ち，ノード u の通信半径を $r_u \in \mathbb{R}_{\geq 0}$ とすると，時刻 t において，$\mathrm{dist}_t(u, v) \leq r_u$ となるノード u, v についてのみ，有向辺 $(u, v) \in E_t$ が存在する．

今回は，全てのノードの通信半径は同じであると仮定し，任意の $u \in V$ について $r_u = r$ とする．この仮定から有向辺 (u, v) が存在することは有向辺 (v, u) が存在することの必要十分条件となり，この二つの有向辺の組 $((u, v), (v, u))$ を一つの無向辺 e_{uv} として扱う．以下では，ネットワークを無向グラフとして扱う．

ノード u との間に無向辺 e_{uv} を持つようなノード v を，ノード u の隣接ノードと呼び，関数 $\Gamma_t : V \to 2^V$ を，時刻 t における，ノード $u \in V$ の隣接ノードを得る関数とする．つまり，$\Gamma_t(u) = \{v \in V | e_{uv} \in E_t\}$ である．

3　不正確な大域情報を利用したブロードキャストアルゴリズム

ブロードキャスト問題とは，あるソースノード $s \in V$ が発信するメッセージを，ネットワーク上の全てのノード $u \in V$ へ伝達するという問題である．ブロードキャスト問題を解くアルゴリズムをブロードキャストアルゴリズムと呼ぶ．

アルゴリズムの実行に用いる不正確な大域情報を，アルゴリズム開始時の全てのノードの位置とする．つまり，全てのノード $v_1, v_2, \cdots, v_n \in V$ について，$\mathrm{pos}_0(v_1), \mathrm{pos}_0(v_2), \cdots, \mathrm{pos}_0(v_n)$ が初期知識として各ノードに与えられると仮定する．この初期位置に関する大域情報から，各ノードは，初期状況における各ノードのソースノードからの距離をローカルに計算することができ，また，各ノードの通信半径は同じであると仮定しているため，ソースノードを根とする全域木や，ソースノードからのホップ数などもローカルに計算することができる．

2.1 節で述べたようなネットワークでは，ある時刻 t に，グラフ G_t が非連結になってしまう可能性があるが，ブロードキャストを可能とするために，任意の時刻 t においてグラフ G_t が連結である場合についてのみシミュレーション実験を行う．また，ノードは受信したメッセージを保持するための記憶領域を持たず，受信したメッセージはすぐに隣接ノードへ転送し，また，1 度しかメッセージの送信は行わないものとする．

3.1　アルゴリズム flooding

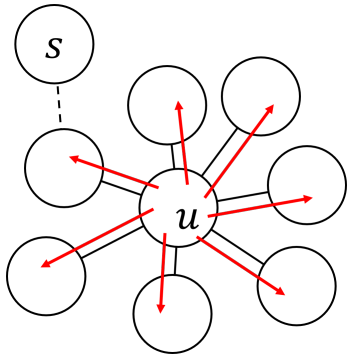

図 1: アルゴリズム flooding が転送する隣接ノード

提案アルゴリズムの比較対象として，従来からよく利用されているフラディングアルゴリズムを示す．このアルゴリズムは，大域情報を利用していない．ソースノードはアルゴリズム実行開始後すぐにメッセージを送信し，他の各ノードは，全ての隣接ノードへ，受信したメッセージを転送する．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，全ての隣接ノード $v \in \Gamma_t(u)$ へ，そのメッセージを転送する．

図 1 に，ノード u が受信したメッセージを転送する隣接ノードについての図を示す．

このアルゴリズムは，トポロジが変化しない静的ネットワークでは，すべてのノードにメッセージが伝達されることを保証する．しかし，各ノードはメッセージを（そのときの全隣接ノードに）一度しか送信しないので，トポロジが変化する動的ネットワークでは，すべてのノードにメッセージが伝達できるとは限らない．

アルゴリズム flooding は大域情報は用いておらず，他の不正確な大域情報を用いたアルゴリズムを評価するための指標として，シミュレーションを行う．この手法は必要以上にメッセージを送信してしまうため，メッセージ量が多くなるが，最も多くのノードにメッセージを伝達できることが予想される．

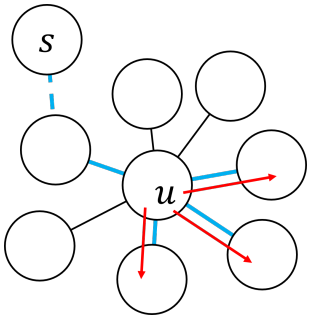

図 2: bft が転送する隣接ノード

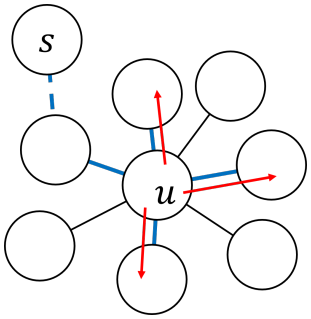

図 3: mst が転送する隣接ノード

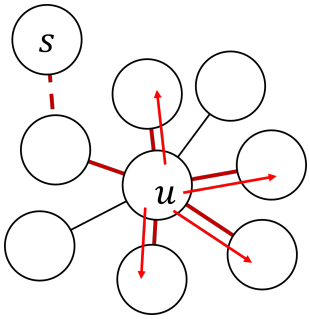

図 4: bftmst が転送する隣接ノード

3.2 アルゴリズム bft

各ノードは初期状況の全てのノードの位置をもとに，ソースノードを根とする共通の幅優先木 T^{bft} をローカルに計算する．ソースノードはアルゴリズム実行開始後すぐにメッセージを送信し，他の各ノードは，T^{bft} の子である隣接ノードへ，受信したメッセージを転送する．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，初期状況におけるソースノードを根とする幅優先木の子を得る関数を $\mathrm{child}_0^{\mathrm{bft}} : V \to 2^V$ とすると，隣接ノード $v \in \Gamma_t(u) \cap \mathrm{child}_0^{\mathrm{bft}}(u)$ へ，そのメッセージを転送する．

図 2 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．太い辺は，初期状況におけるソースノードを根とする幅優先木を表し，ノード u はその幅優先木に沿ってメッセージを転送する．

静的ネットワークではこの手法で，フラディングよりも少ないメッセージ量でブロードキャストを行うことができる．ただし，動的アドホックネットワークでは，ノードの移動により辺が変化し，メッセージ送信時に T^{bft} の子ノード全てと隣接しているとは限らないので，アルゴリズム bft ではメッセージを伝達できないノードが多く存在することが予想される．幅優先木は，ソースノードからのホップ数を最小にするため，初期状況で長い辺を用いる可能性が高く，移動に対する辺の耐性は高くないが，ソースノードからのホップ数が最小であるため，子の経路が移動の影響を受けにくい可能性もある．

3.3 アルゴリズム mst

各ノードは初期状況の全てのノードの位置をもとに，ノード間の距離を重みとした，ソースノードを根とする共通の最小重み全域木 T^{mst} をローカルに計算する．ソースノードはアルゴリズム実行開始後すぐにメッセージを送信し，他の各ノードは，T^{mst} の子である隣接ノードへ，受信したメッセージを転送する．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，初期状況におけるソースノードを根とする最小重み全域木の子を得る関数を $\mathrm{child}_0^{\mathrm{mst}} : V \to 2^V$ とすると，隣接ノード $v \in \Gamma_t(u) \cap \mathrm{child}_0^{\mathrm{mst}}(u)$ へ，そのメッセージを転送する．

図 3 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．太い辺は，初期状況におけるソースノードを根とする最小重み全域木を表し，ノード u はその最小重み全域木に沿ってメッセージを転送する．

静的ネットワークでは，幅優先木を用いる手法同様，この手法で，フラディングよりも少ないメッセージ量でブロードキャストを行うことができる．最小重み全域木は，初期状況で短い辺 (重みの小さい辺) を用いるので，移動に対する辺の耐性は高くなるが，ソースノードからのホップ数が大きくなりがちであり，この経路が移動の影響を受けやすい可能性もある．

3.4 アルゴリズム bftmst

この手法では，幅優先木とノード間の距離を重みとする最小重み全域木の両方を利用する．ソースノードはアルゴリズム開始後すぐにメッセージを送信し，他の各ノードは，T^{bft} または T^{mst} の子である隣接ノードへ，受信したメッセージを転送する．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，隣接ノード $v \in \Gamma_t(u) \cap (\mathrm{child}_0^{\mathrm{bft}}(u) \cup \mathrm{child}_0^{\mathrm{mst}}(u))$ へ，そのメッセージを転送する．

図 3 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．太い辺は，初期状況におけるソースノードを根とする幅優先木および最小重み全域木を表し，ノード u はその辺に沿ってメッセージを転送する．

静的ネットワークでは，幅優先木を用いる手法同様，この手法で，フラディングよりも少ないメッセージ量でブロードキャストを行うことができる．一方，動的アドホックネットワークでは，ブロードキャストを行えることは保証できない．幅優先木と最小重み全域木の両方を用いるため，アルゴリズム bftmst， mst よりも多くのメッセージを用いるが，より多くのノードにメッセージを伝達できると期待される．

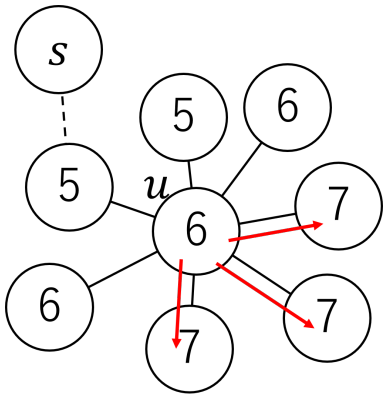

図 5: アルゴリズム gthop が転送する隣接ノード

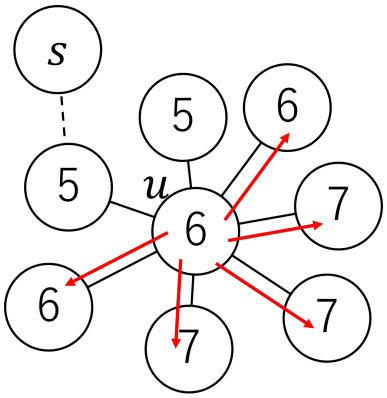

図 6: アルゴリズム hop が転送する隣接ノード

3.5 アルゴリズム gthop

ソースノードはアルゴリズム実行開始後すぐにメッセージを送信し，他の各ノードは，初期状況でのソースノードからの最小ホップ数が自身の最小ホップ数以上である隣接ノードへ，受信したメッセージを転送する．つまり，初期状況におけるソースノードからのホップ数を得る関数を $\mathrm{hop}_0 : V \to \mathbb{N}$ とすると，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，式 (1) で表される隣接ノード $v \in \Gamma_t^{\mathrm{gthop}}(u)$ へ，そのメッセージを転送する．

$$\Gamma_t^{\mathrm{gthop}}(u) = \{v | (v \in \Gamma_t(u)) \wedge (\mathrm{hop}_0(u) < \mathrm{hop}_0(v))\} \tag{1}$$

図 5 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．各ノードの数字はソースノードからの初期状況におけるホップ数を表す．

静的ネットワークではこの手法で，フラディングよりも少ないメッセージ量でブロードキャストを行うことができる．この手法で用いる辺は，アルゴリズム bft で用いる辺を含むので，アルゴリズム bft よりも多くのメッセージを必要とするが，より多くのノードにメッセージを伝達できると期待できる．

3.6 アルゴリズム hop

アルゴリズム gthop を，最小ホップ数が等しい隣接ノードにもメッセージを送信するように変更したものである．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，式 (2) で表される隣接

ノード $v \in \Gamma_t^{\mathrm{hop}}(u)$ へ，そのメッセージを転送する．

$$\Gamma_t^{\mathrm{hop}}(u) = \{v | (v \in \Gamma_t(u)) \wedge (\mathrm{hop}_0(u) \leq \mathrm{hop}_0(v))\} \tag{2}$$

図 6 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．各ノードの数字はソースノードからの初期状況におけるホップ数を表す．

静的ネットワークではこの手法で，フラディングよりも少ないメッセージ量でブロードキャストを行うことができる．また，この手法はアルゴリズム gthop よりもメッセージ量は大きくなるが，アルゴリズム gthop より多くのノードにメッセージを伝達できることが期待できる．

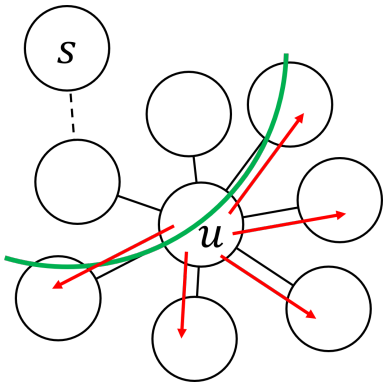

図 7: アルゴリズム far が転送する隣接ノード

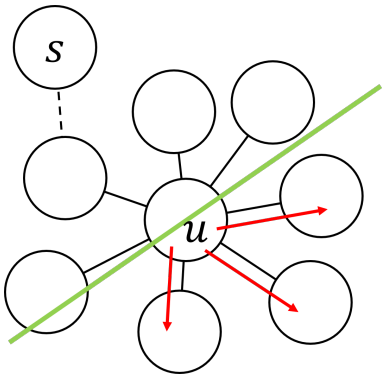

図 8: アルゴリズム area が転送する隣接ノード

3.7 アルゴリズム far

ソースノードはアルゴリズム実行開始後すぐにメッセージを送信し，他の各ノードは，初期状況でのソースノードからのユークリッド距離が，自身の初期状況でのソースノードからのユークリッド距離よりも大きい隣接ノードへ，受信したメッセージを転送する．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，式 (3) で表される隣接ノード $v \in \Gamma_t^{\mathrm{far}}(u)$ へ，そのメッセージを転送する．

$$\Gamma_t^{\mathrm{far}}(u) = \{v | (v \in \Gamma_t(u)) \wedge (\mathrm{dist}_0(s,u) < \mathrm{dist}_0(s,v))\} \tag{3}$$

図 7 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．図の太い円弧は，ソースノードを中心とし，u を通る円の一部を表す．

上記の 6 つの手法とは異なり，アルゴリズム far は，静的ネットワークでも，ブロードキャストを行えることを保証できない．

3.8 アルゴリズム area

初期状況でのソースノードと自身を結ぶ線分に垂直な，自身を通る直線でフィールドを 2 つの領域に分割する．この 2 つの領域のうち，ソースノードを含まない方の領域に含まれる隣接ノードへ，各ノードは受信したメッセージを送信する．ただし，ソースノードは全ての隣接ノードへメッセージを送信する．つまり，任意のノード $u \in V$ が，時刻 t に初めてメッセージを受信した場合，式 (4) で表される隣接ノード $v \in \Gamma_t^{\mathrm{area}}(u)$ へ，そのメッセージを転送する．

$$\Gamma_t^{\mathrm{area}}(u) = \left\{ v \middle| (v \in \Gamma_t(u)) \wedge \left(\frac{\pi}{2} < \arccos \left(\frac{(\mathrm{pos}_0(s) - \mathrm{pos}_0(u)) \cdot (\mathrm{pos}_0(v) - \mathrm{pos}_0(u))}{\mathrm{dist}_0(s,u)\,\mathrm{dist}_0(v,u)} \right) < \frac{3\pi}{2} \right) \right\} \tag{4}$$

図 8 に，ノード u が受信したメッセージを転送する隣接ノードの例を示す．図の太い直線は，ソースノードと u を結ぶ線分に垂直な，u を通る直線を表す．

この手法もアルゴリズム far と同様に，静的ネットワークでもブロードキャストを行えることができない可能性がある．また，アルゴリズム area はアルゴリズム far よりもメッセージ量が少なくなる．

4　シミュレーション結果

シミュレーションは 1000×1600 のフィールドおノードが移動する環境に対して行う．通信半径はノード数にかかわらず 250 とした．ノード数は $50, 100, 150, 200, 250, 300$ について行い，フィールド内でノードが疎な場合から密な場合にかけて，シミュレーションを行った．このとき，3 節で述べたように，グラフが常に連結であった場合のみを対象とした．

4.1　アルゴリズムの完全性

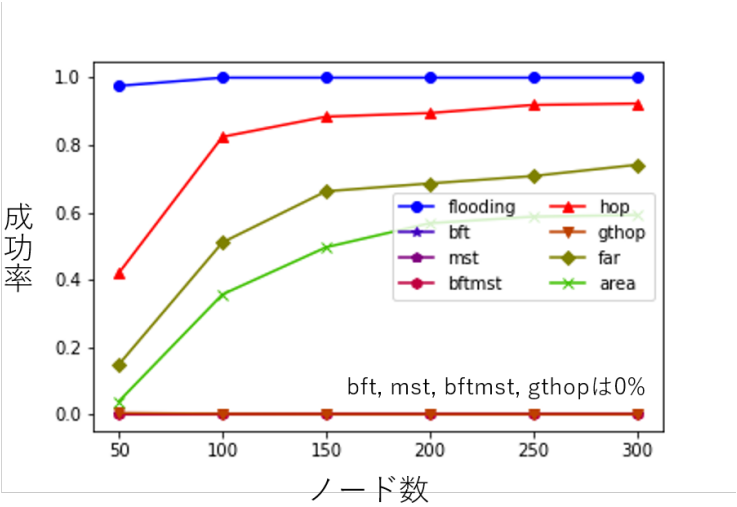

図 9: ブロードキャストの成功率

図 9 に，ノード数ごとのブロードキャスト成功率を示す．ブロードキャスト成功率とは，行ったシミュレーションのうち，全てのノードにメッセージが伝達しブロードキャストが成功した割合を指す．

アルゴリズム flooding については静的ネットワークと同様，1 に近い成功率を得た．アルゴリズム bft や mst や bftmst は静的ネットワークでのブロードキャストは保証されていたが，動的アドホックネットワークではシミュレーションを行った全てのノード数についてブロードキャスト成功率はほぼ 0 となった．アルゴリズム gthop や hop も，静的ネットワークでのブロードキャストや保証されていた．gthop の成功率はほぼ 0 となった一方で，hop は 4 から 8 割の成功率となった．アルゴリズム far や area は，静的ネットワークでのブロードキャストは保証されていなかった．ノード数が少なく，ネットワークが疎である場合は成功率はほぼ 0 となったのに対し，ノード数が増加し，ネットワークが密になるにしたがって，成功率は 6 割ほどまで上昇した．これは，ネットワークが密な場合は，より多くの辺が伝達に用いられるためだと考えられる．

図 10 に，ノード数ごとの受信したノード数の平均値を，図 11 に，それをノード数で割ったものを示す．

アルゴリズム bft や mst や bftmst 以外の，木を用いない手法については，多くのノードへメッセージを伝えることができていた．一方で木を用いる手法については，mst はあまり多くのノードにメッセージを伝えることはできず，bft は mst よりは多くのノードにメッセージを伝えることができていた．この 2 つを組み合わせた bftmst は，そのどちらよりも多くのノードにメッセージを伝えることができており，木を用いない手法に迫る受信率となった．

4.2　アルゴリズムの効率

図 12 に，ノード数ごとの受信されたメッセージ総数の平均をノード数ごとに示す．

フィールドと通信半径がノード数に関わらず一定であることから，各ノードの次数（隣接ノードの数）の平均はノード数にほぼ比例する．つまり，アルゴリズム bft や mst や bftmst 以外の手法では，メッセージ数はノード数の 2 乗にほぼ比例すると考えられる．実際に，図 12 からメッセージ数がノード数の 2 乗にほぼ比例していることがわかる．

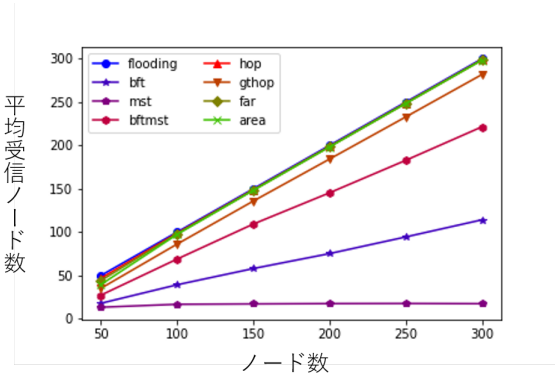

図 10: 受信したノードの数

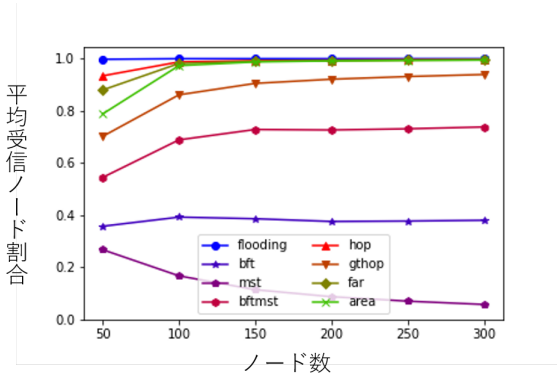

図 11: 受信したノードの割合

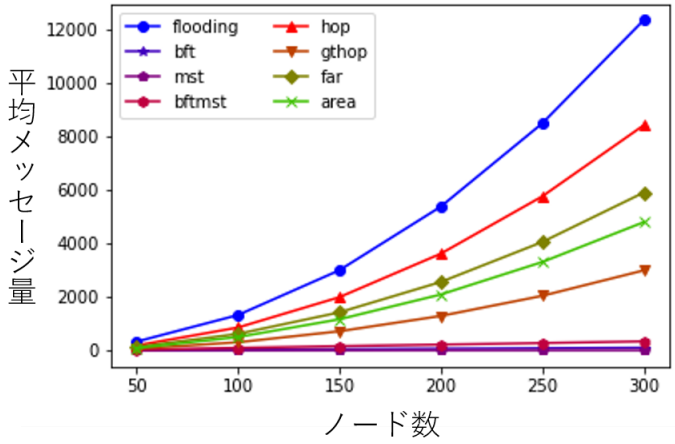

図 12: 受信メッセージ数

対して，アルゴリズム bft や mst は，静的ネットワークでは木の辺数が $n-1$ であることから，メッセージ数は最大でも $n-1$ となるため，フラディングよりも漸近的に非常に小さいメッセージ数となる．これはアルゴリズム bftmst についても同様で，メッセージ数は最大でも $2(n-1)$ となる．実際に図 12 ではアルゴリズム bft や mst や bftmst のメッセージ数は他のアルゴリズムよりもはるかに小さくなっている．

3.8 節で示したように，アルゴリズム area の平均メッセージ量はアルゴリズム far の平均メッセージ量よりも少なかった．

さらに，アルゴリズム area の平均メッセージ量は，アルゴリズム flooding の平均メッセージ量のおよそ半分となった．これについて，アルゴリズム area の定義から平均メッセージ量はアルゴリズム flooding のおよそ半分となることが予想され，実際に，そのようになっていることがわかった．

一方で，アルゴリズム gthop や hop の平均メッセージ量は，アルゴリズム far と同程度と予想されたが，図 12 のように予想に反した結果が得られた．

5　まとめ

本稿では不正確な大域情報を用いてブロードキャスト問題を解くアルゴリズムをいくつか提案し，平面上を移動するノードから構成される動的アドホックネットワークにおけるシミュレーションによって性能評価を行った．

今回の研究で，大域情報は不正確でも，分散アルゴリズムの効率化に有用であることが実証できた．今

後の課題として，シミュレーションの設定の変化が結果にどのような影響を与えるかを検証したい．また，ブロードキャスト以外の他の問題についても，どのような大域情報が，またその不正確さの程度が分散アルゴリズムの効率や解の正確性にどのような影響を与えるかを実験的に，可能であれば解析的に，解明したい．

謝辞　本研究の一部は JSPS 科研費 19H04085 および 20H04140 の助成を受けたものです．

参考文献

[1] Gerard Tel. *Introduction to distributed algorithms*. Cambridge university press, 2000.

[2] Michel Raynal. *Distributed algorithms for message-passing systems*. Springer, 2013.

[3] Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004.

[4] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, Vol. 8, No. 2-3, pp. 153–167, 2002.

[5] Min-Te Sun, Wuchi Feng, and Ten-Hwang Lai. Location aided broadcast in wireless ad hoc networks. In *GLOBECOM'01. IEEE Global Telecommunications Conference*, Vol. 5, pp. 2842–2846. IEEE, 2001.

[6] Wei Peng and Xicheng Lu. AHBP: An efficient broadcast protocol for mobile ad hoc networks. *Journal of Computer Science and Technology*, Vol. 16, No. 2, pp. 114–125, 2001.

[7] Young-Bae Ko and Nitin H Vaidya. Location-Aided Routing (LAR) in mobile ad hoc networks. *Wireless Networks*, Vol. 6, No. 4, pp. 307–321, 2000.

[8] Mohammad A Mikki. Energy efficient location aided routing protocol for wireless MANETs. *arXiv preprint arXiv:0909.0093*, 2009.

[9] Hyojun Lim and Chongkwon Kim. Flooding in wireless ad hoc networks. *Computer Communications*, Vol. 24, No. 3-4, pp. 353–363, 2001.