アドホックネットワークのための中継ログ手法を用いたチェックポイントプロトコル

小野真和[†1] 桧垣博章[†2]

分散コンピューティング環境において耐故障性を実現する手法として，従来の固定ネットワーク環境ではチェックポイントリカバリプロトコルが提案されてきた．チェックポイントリカバリプロトコルでは，状態情報を格納する安定記憶の存在と，メッセージの送信元コンピュータと送信先コンピュータの同期による一貫性のないメッセージ（紛失メッセージと孤児メッセージ）の検出，回避が十分に可能な通信帯域の存在が前提となっている．本論文では，これらの前提条件が成立しない，移動コンピュータのみで構成されるアドホックネットワークのためのチェックポイントリカバリ手法を提案する．提案手法では，移動コンピュータの状態情報を隣接移動コンピュータに保存する．また，紛失メッセージとなる可能性のあるメッセージを無線マルチホップ配送経路上の中継移動コンピュータが判定してメッセージログに保存する中継ログ方式を用いることとする．このメッセージログとチェックポイントにおける状態情報は，チェックポイント設定要求メッセージにピギーバックして配送する．提案手法は，必要とするメッセージ数，メッセージ量が多くの場合従来手法よりも小さくなる．

Checkpoint Protocol with Intermediate Message Logging for Mobile Ad-hoc Networks

Masakazu Ono[†1] and Hiroaki Higaki[†2]

For achieving mission-critical network applications, checkpoint recovery protocols have been researched and developed. In coventional protocols for wired networks, stable storages to store state information are assumed and enough bandwidth is assigned to synchronize a source and a destination computers of a message in order to avoid that the message becomes inconsistent, i.e. neither orphan nor lost. In this paper, we propose a novel checkpoint protocol in ad hoc networks without stable storages and enough communication bandwidth. Here, a checkpoint request message is delivered by flooding. State information of a mobile computer is carried by this message and stored into neighbor mobile computers. A candidate of a lost message is detected and stored by an intermediate mobile computer on its transmission route. Here, communication overhead for taking a global checkpoint is reduced.

1. はじめに

ノート型コンピュータや PDA などの移動コンピュータを IEEE802.11[26] や HIPERLAN[10]，Bluetooth[27] などの無線通信プロトコルを用いて相互に接続した無線 LAN 技術の研究開発が進み，その使用が普及している．また，無線基地局を介して有線ネットワークと接続されたインフラストラクチャネットワークだけでなく，移動コンピュータだけで構成されるアドホックネットワークへの要求が高まっている．アドホックネットワークの適用例として，一時的に構築されるイベント会場や災害現場などでのネットワーク，危険地帯で無線基地局が設置できない場所における自律移動型ロボットの協調動作のためのネットワーク，センサネットワーク[5),17)] などがある．このようなアドホックネットワークにおけるミッションクリティカルアプリケーションの実行を考えたとき，耐故障性を実現するためのチェックポイントリカバリ手法を適用することが考えられる．

しかし，有線ネットワーク環境を対象とした従来のチェックポイント手法[11),14)] は安定記憶[16)] が各コンピュータに存在することを前提としている．また，一貫性のないメッセージ（孤児メッセージと紛失メッセージ）を送信元コンピュータと送信先コンピュータの同

†1 東京電機大学大学院先端科学技術研究科
Graduate School of Advanced Science and Technology, Tokyo Denki University
†2 東京電機大学未来科学部ロボット・メカトロニクス学科
Department of Robotics and Mechatronics, Tokyo Denki University

期メッセージの交換によって検出することが可能となる十分な帯域幅がネットワークによって提供されているとしている．そのため，アドホックネットワーク上の移動コンピュータは安定記憶を持つことができないという問題や，アドホックネットワークにおける隣接移動コンピュータ間の無線通信リンクが狭帯域幅で低信頼であるため同期メッセージ交換に要する通信オーバヘッドが大きいという問題がある．本論文では，アドホックネットワークにおいて安定記憶を実現し，一貫性のないメッセージの発生を回避するための送受信コンピュータ間における同期メッセージ交換に起因する通信オーバヘッドを削減する新たなチェックポイントプロトコルを提案する．

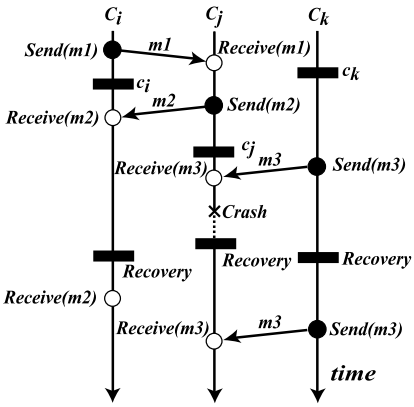

図 1 チェックポイントリカバリ
Fig. 1 Checkpoint recovery.

2. 従 来 手 法

2.1 チェックポイントプロトコルとメッセージログ

一般に，コンピュータネットワーク $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ は，ネットワークインタフェースを介してネットワークに接続されたコンピュータの集合 $\mathcal{V}$ と各コンピュータ $C_i \in \mathcal{V}$ から他のコンピュータ $C_j \in \mathcal{V}$ への通信リンク $|C_i, C_j\rangle$ の集合 $\mathcal{E}$ からなる．各コンピュータ C_i のアプリケーションでは，メッセージ送信イベント $Send(m)$，メッセージ受信イベント $Receive(m)$，および通信をともなわないローカルイベントが生起し，それとともに C_i の状態が変化する．

これまでに，コンピュータの障害に対して耐性を持つアプリケーション実行環境の実現には空間的および時間的な冗長性を導入することが検討されてきた．空間的冗長性を導入する手法が複製方式である．ここでは，各コンピュータを複数の複製コンピュータとして実現し，複製コンピュータの状態を同一に保つことによって，一部の複製に障害が発生してもアプリケーション実行を継続させる．一方，時間的冗長性を導入する手法が本論文で対象とするチェックポイント方式である．

チェックポイント方式では，**図 1** に示すように，正常に動作している各コンピュータ C_i が自身の状態情報を安定記憶に保存することによってローカルチェックポイント c_i を設定し，いずれかのコンピュータ C_j が障害から回復した場合には，すべてのコンピュータが安定記憶に保存したローカルチェックポイントの状態を復元することによってアプリケーションを再開する．ネットワークに接続されたコンピュータ間のメッセージ配送には一定ではない遅延を要することから，各コンピュータが自身の時計を参照することによって同時にローカルチェックポイントを設定することは困難である．そこで，各コンピュータの設定するローカルチェックポイント c_i の集合であるグローバルチェックポイント $\mathcal{C}_\mathcal{V}$ における各コンピュータの状態が一貫性を持つように c_i を設定するためのチェックポイントプロトコルが多数提案されている[9]．ここでは，一貫性を持つグローバルチェックポイントは，以下のように定義される．

[定義：イベント間の因果先行関係]
イベント e がイベント e' に因果先行する（$e \rightarrow e'$）とは，以下のいずれかの条件を満足することである．

- 同一のコンピュータで e は e' よりも先に生起する．
- e はアプリケーションにおけるメッセージ m の送信イベント $Send(m)$ であり，e' は m の受信イベント $Receive(m)$ である．なお，以降の図では，$Send()$ と $Receive()$ のそれぞれを●と○で表記する．
- あるイベント e'' について，$e \rightarrow e''$ かつ $e'' \rightarrow e'$ である． □

[定義：紛失メッセージ]
メッセージ m が送信元コンピュータ C_s から送信先コンピュータ C_d へ配送され，グローバルチェックポイント $\mathcal{C}_\mathcal{V}$ における C_s のローカルチェックポイント $c_s \in \mathcal{C}_\mathcal{V}$ と C_d のローカルチェックポイント $c_d \in \mathcal{C}_\mathcal{V}$ に対して $Send(m) \rightarrow c_s$ かつ $c_d \rightarrow Receive(m)$ であるならば，m は $\mathcal{C}_\mathcal{V}$ の紛失メッセージである（**図 2**）． □

[定義：孤児メッセージ]
メッセージ m が送信元コンピュータ C_s から送信先コンピュータ C_d へ配送され，グローバルチェックポイント $\mathcal{C}_\mathcal{V}$ における C_s のローカルチェックポイント $c_s \in \mathcal{C}_\mathcal{V}$ と C_d のローカルチェックポイント $c_d \in \mathcal{C}_\mathcal{V}$

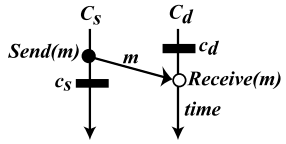

図 2 紛失メッセージ
Fig. 2 Lost message.

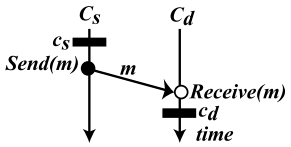

図 3 孤児メッセージ
Fig. 3 Orphan message.

に対して $c_s \rightarrow Send(m)$ かつ $Receive(m) \rightarrow c_d$ であるならば，m は $\mathcal{C}_{\mathcal{V}}$ の孤児メッセージである（**図 3**）．□

ここで，通信リンク $|C_s, C_d\rangle$ を配送されるメッセージ m が孤児メッセージであるならば，c_s と c_d を含むグローバルチェックポイント $\mathcal{C}_{\mathcal{V}}$ は一貫性を持たない．一方，m が紛失メッセージであるならば，$\mathcal{C}_{\mathcal{V}}$ における m は $|C_s, C_d\rangle$ を配送途中のメッセージと見なすことができる．このため，m の存在によって $\mathcal{C}_{\mathcal{V}}$ の一貫性が損なわれることはない．ただし，リカバリ後に C_s は c_s からアプリケーションを再開することから，$Send(m) \rightarrow c_s$ である $Send(m)$ が再度生起することはない．そこで，リカバリ後に C_d において $Receive(m)$ が生起し，アプリケーションが m を受理可能としなければならない．たとえば，図 1 では，リカバリ後に $Receive(m_2)$ で C_i が m_2 を受理できなければならない．

[定義：一貫性を持つグローバルチェックポイント] グローバルチェックポイント $\mathcal{C}_{\mathcal{V}}$ の孤児メッセージが存在せず，すべての紛失メッセージがリカバリ後にアプリケーションで受理可能であるとき，$\mathcal{C}_{\mathcal{V}}$ は一貫性を持つグローバルチェックポイントである．□

従来のチェックポイントプロトコルでは，孤児メッセージの発生はコンピュータ間の同期メッセージの交換によって回避されている．Koo らのプロトコル[15]では，チェックポイントプロトコルの指揮コンピュータ C_0 と他のコンピュータとの間でチェックポイント設定要求メッセージ *Creq*，チェックポイント設定応答メッセージ *Crep*，チェックポイント設定終結メッセージ *Cfin* が **図 4** のように交換され，各コンピュータ C_i における *Creq* の受信から *Cfin* の受信までの間，送信イベント $Send(m)$ の生起を禁止することによって孤児メッセージの発生を回避している．

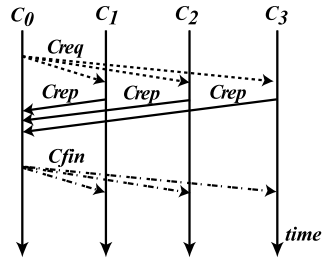

図 4 Koo らのチェックポイントプロトコル
Fig. 4 Koo's checkpoint protocol.

一方，紛失メッセージをリカバリ後に送信先コンピュータのアプリケーションが受理可能とするための手法として，メッセージをその送信元コンピュータで保存する送信ログ方式[1),2),12),28)] と送信先コンピュータで保存する受信ログ方式[3),4),13),24),25),29)] が提案されている．紛失メッセージの定義から，通信リンク $|C_s, C_d\rangle$ を配送されるメッセージ m が紛失メッセージであることを判定できるのは C_s ではなく C_d である．これは，$c_s, c_d \in \mathcal{C}_{\mathcal{V}}$ について，$Send(m) \rightarrow c_s$ であることを C_s が m に記録し，C_d が $c_d \rightarrow Receive(m)$ を満足することを確認することで実現できる．したがって，受信ログ方式では紛失メッセージのみをメッセージログに保存することが可能である．これに対して，送信ログ方式では，m が紛失メッセージになるか否かを $Send(m)$ において C_s が特定できないことから，すべてのメッセージをログに保存し，リカバリ時に必要なものを選択して再配送しなければならない．ただし，送信ログ方式では，紛失メッセージは送信元コンピュータではローカルチェックポイント設定前に送信されていることから，コンピュータの状態情報とともにメッセージログを安定記憶へ保存することができる．これに対して，受信ログ方式では，紛失メッセージは送信先コンピュータではローカルチェックポイントの設定後に受信されることから，すべての紛失メッセージが送信先コンピュータに配送されたことを確認する手段が必要である．Chandy らのプロトコル[7]では，通信リンクが FIFO であることを前提にして，すべての通信リンクに沿って同期メッセージを配送し，この配送の終了後にコンピュータの状態情報とメッセージログを安定記憶へ保存する（**図 5**）．

2.2 無線アドホックネットワークと無線マルチホップ配送

無線アドホックネットワーク $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ は，移動コンピュータの集合 $\mathcal{V}$ と各移動コンピュータ $M_i \in \mathcal{V}$ からその無線信号到達範囲内にある隣接移動コンピュー

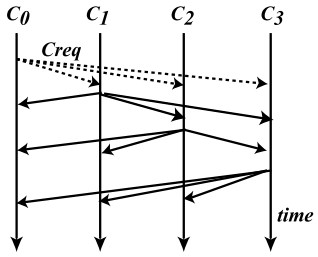

図 5 Chandy らのチェックポイントプロトコル
Fig. 5 Chandy's checkpoint protocol.

タ M_j への無線通信リンク $|M_i, M_j\rangle$ の集合 $\mathcal{E}$ からなる．本論文では，すべての移動コンピュータ M_i の無線信号到達範囲の形状は M_i を中心とする同一半径の円であると仮定する．したがって，すべての隣接移動コンピュータ対の間は双方向無線通信リンクによって接続されている．すなわち，$|M_i, M_j\rangle \in \mathcal{E}$ ならば $|M_j, M_i\rangle \in \mathcal{E}$ である．

送信元移動コンピュータ M_s（$= M_0$）から送信先移動コンピュータ M_d（$= M_n$）へメッセージ m を配送するとき，無線通信リンク $|M_s, M_d\rangle$ が存在しないならば，m の無線マルチホップ配送が適用される．すなわち，$|M_i, M_{i+1}\rangle \in \mathcal{E}$（$i = 0, \ldots, n-1$）を満足する移動コンピュータ列からなる無線マルチホップ配送経路 $||M_0, \ldots, M_n\rangle\rangle$ に沿った m の配送がなされる．送信元移動コンピュータ M_0 のアプリケーションで $Send(m)$ イベントが生起すると，送信イベント $send(m)$ によってメッセージ m が隣接移動コンピュータの 1 つである次ホップ移動コンピュータ M_1 へと転送される．中継移動コンピュータ M_i は，前ホップ移動コンピュータ M_{i-1} から転送されたメッセージ m を受信イベント $receive(m)$ で受信し，受信した m を送信イベント $send(m)$ によって次ホップ移動コンピュータ M_{i+1} へと転送する．こうして，送信先移動コンピュータ M_n が前ホップ移動コンピュータ M_{n-1} から転送された m を受信イベント $receive(m)$ で受信すると，M_n のアプリケーションが受信イベント $Receive(m)$ によって受理することが可能となる．

2.3 無線アドホックネットワークにおけるチェックポイントプロトコルの問題点

移動コンピュータネットワークにおいてチェックポイントリカバリを実現するプロトコルの研究例には文献 1)，2)，6)，18)–23)，29) などがある．ここでは，送信元移動コンピュータと送信先移動コンピュータが直接メッセージを交換することが可能である場合と，すべての移動コンピュータが固定基地局とのみ直接メッセージを交換する場合，すなわち送信元移動コンピュータから送信先移動コンピュータへ基地局を介してメッセージを配送する場合について検討されている．本論文では，無線アドホックネットワークにおけるチェックポイントリカバリの適用を考えるが，チェックポイントプロトコルの設計においては，以下の問題点を解決する必要がある．

アドホックネットワークを構成する無線通信リンクは，有線通信リンクよりも帯域幅が狭く，無線信号の減衰と複数無線信号の衝突による低信頼性，無線通信リソースに対する複数の予約の間の競合とマルチホップ配送による遅延の拡大といった特性により，通信に要するオーバヘッドが大きい．このため，チェックポイントプロトコルにおいて，多数の制御メッセージ交換によって移動コンピュータ間の同期を実現し，孤児メッセージの発生回避や紛失メッセージの検出を行うことは困難である．したがって，制御メッセージ数の少ないプロトコルを設計することが求められる．アドホックネットワークで受信ログ方式を用いて紛失メッセージをメッセージログに保存する手法では，すべての紛失メッセージをその送信先移動コンピュータに受信させるために，Chandy らのプロトコルに示す方法ですべての移動コンピュータ対の間で同期メッセージを交換することが必要である．アドホックネットワークでは，この同期メッセージをすべての移動コンピュータ間のマルチホップ配送経路に沿って交換することになり，その通信オーバヘッドは大きい．また，同じ理由により，メッセージ量の小さなプロトコルを設計することが求められる．

アドホックネットワークを構成する移動コンピュータは，電源容量，CPU 計算能力，メモリ容量，2 次記憶容量といった計算リソースも必ずしも十分ではない．このため，多数のメッセージをメッセージログに保存する方法の適用も困難である．アドホックネットワークで送信ログ方式を用いる手法では，ローカルチェックポイント設定以前に生起した送信イベントによって送信されたすべてのメッセージを送信元コンピュータが紛失メッセージになりうるメッセージとして保存するため，要求される記憶容量が大きい．さらに，携帯性，移動性による不安定な環境での使用となるため，単一の移動コンピュータに故障独立な複数の 2 次記憶装置による安定記憶を実現することは困難である．

また，アプリケーションの停止は極力避けることが必要である．あるイベント生起を待って他のイベントの生起を遅延させることが必要となる場合においても，その待ち時間を短縮することが求められる．

3. 提案手法

前章で述べたアドホックネットワークの特徴に基づく問題点を解決したチェックポイント手法を提案する．本論文で提案する手法は，Koo らのプロトコル[15]におけるチェックポイント設定要求メッセージ *Creq*，チェックポイント設定応答メッセージ *Crep*，チェックポイント設定終結メッセージ *Cfin* の交換による 3 フェーズの手法に基づくものとする．*Creq*，*Cfin* はフラッディング配送を行い，*Crep* は *Creq* のフラッディング配送で形成されたスパニングツリーを用いて配送する．ただし，Koo らのプロトコルと異なり，孤児メッセージの発生を回避するために，各移動コンピュータにおける *Creq* の受信から *Cfin* の受信までの間の送信イベント生起を禁止することはしない．

各移動コンピュータにおけるチェックポイント設定時の状態情報とメッセージログは，その移動コンピュータ自身とその隣接移動コンピュータに保存する．この方式は，単体では容量，安定性ともに不十分な移動コンピュータ群において，複数の移動コンピュータの 2 次記憶を用いることで分散型の簡易安定記憶を実現する．各移動コンピュータの状態情報とメッセージログはフラッディング配送される *Creq* にピギーバックして隣接移動コンピュータへ配送する．

紛失メッセージの検出手法として，マルチホップ配送経路上のすべての移動コンピュータにおいて紛失メッセージの検出とメッセージログへの保存を行う中継ログ方式を提案する．中継移動コンピュータは，転送処理を行うマルチホップ配送中のメッセージ m が紛失メッセージとなる可能性があるかを判定し，その可能性がある場合にはメッセージログに保存する．この方式を用いることによって，各移動コンピュータがチェックポイントにおいて保存するべきメッセージログを *Creq* のブロードキャスト送信以前に決定することができる．ただし，2.1 節の定義により，最終的に m が紛失メッセージとなるかを判定できるのは m の送信先移動コンピュータのみである．そのため，中継移動コンピュータのメッセージログに保存されたにもかかわらず，紛失メッセージとはならないメッセージが存在しうる．提案手法では，このようなメッセージはリカバリ後にメッセージログから除去し，送信先移動コンピュータのアプリケーションに再受理させないようにする．これは，中継移動コンピュータのメッセージログに保存された紛失メッセージとはならないメッセージを送信先移動コンピュータが自身の否定メッセージログに保存することによって実現される．この否定メッセージログも *Creq* のブロードキャスト送信以前に決定することが可能であり，フラッディング配送される *Creq* にピギーバックすることができる．

3.1 前提条件

本論文で提案するアドホックネットワークのためのチェックポイントプロトコルは，以下の条件のもとに構成する．

[前提条件]

1) 隣接する移動コンピュータ間の通信リンクは双方向であり，半二重通信が行われる．また，ユニキャスト通信は，受信確認と再送信機構により，メッセージの紛失なく実現されているものとする．
2) 移動コンピュータ間の通信リンクは，動的に確立および切断されることがある．各移動コンピュータは，隣接移動コンピュータのリストを保持しており，これを更新するプロトコルが適宜動作しているものとする．
3) アドホックネットワークに含まれるすべての移動コンピュータ対は，チェックポイントプロトコルの実行中，マルチホップ配送により互いにメッセージ交換が可能である． □

IEEE802.11 シリーズなどの無線 LAN プロトコルにおいては，隣接移動コンピュータとの直接通信において，受信確認とタイマを用いた再送信機構が導入され，無線通信リンクを介した高信頼な通信を実現している．したがって，前提条件 1) はこのような環境においては自然なものであり，以降の説明においては再送信機構によって紛失のない配送がなされていることを前提にした記述とする．

提案手法では，チェックポイント設定要求メッセージ *Creq* をフラッディングによってアドホックネットワークに属するすべての移動コンピュータに配送する．この *Creq* メッセージは，高信頼に配送されることが必要である．3.2 節で述べるようにすべての隣接移動コンピュータからの *Creq* 受信を確認することで，*Creq* の再送信が不要であることを判定している．これを実現するために，各移動コンピュータは隣接移動コンピュータの集合を保持し，この集合が無線通信リンクの確立，切断を機会に更新されていくことが必要である．いくつかのルーティングプロトコルなどで使用されているように，定期的に無線通信リンクの存在を確認する Hello メッセージの交換を行うことは有効な手法であり，本論文ではこれが機能していることを前提条件 2) としてあげている．

アドホックネットワークにおけるチェックポイントプロトコルの目的は，アドホックネットワークに属す

るすべての移動コンピュータが状態情報を保存することでローカルチェックポイントを設定し，ローカルチェックポイントの集合であるグローバルチェックポイントに一貫性がある，すなわちこれらのすべてのローカルチェックポイント対の間に矛盾がないものとすることである．ここで，チェックポイントプロトコル実行中にアドホックネットワークへの参加，退出が発生すると，すべての移動コンピュータにおいてチェックポイントを設定するための手続きが複雑になる．たとえば，新規にアドホックネットワークに参加した移動コンピュータが存在する場合，フラッディング中の *Creq* を転送することによってこの移動コンピュータでもローカルチェックポイントを設定することが可能である．ただし，アドホックネットワークに属するすべての移動コンピュータは，新規参加移動コンピュータが隣接する場合に備えて *Creq* をバッファに保持することが必要である．一方，チェックポイントプロトコル実行中にアドホックネットワークから退出した移動コンピュータが存在する場合，*Creq* メッセージのすべての移動コンピュータへの到達性が保障できない．以上により，本論文では，アドホックネットワークに属する任意の移動コンピュータ間の到達性がチェックポイントプロトコル実行中は保障されているという前提条件 3) のもとに記述する．なお，この条件を満足するためにはチェックポイントプロトコルの実行に長時間を要することは好ましくない．そこで，チェックポイントプロトコルを実行するための各移動コンピュータにおけるイベントの生起と処理は，他のイベントの生起に対してできるだけブロックされないような構成とすることが求められる．

3.2 フラッディングを用いたチェックポイント手法

本論文で提案するアドホックネットワークのためのチェックポイントプロトコルでは，アドホックネットワークに属するいずれかの移動コンピュータが新しいチェックポイントを設定する必要性を検出し，チェックポイント設定要求メッセージ *Creq* をフラッディング[8]を用いて配送することによって，マルチホップ通信で到達可能なすべての移動コンピュータにその必要性を通知する（**図 6**）．ただし，*Creq* は各移動コンピュータによってブロードキャスト送信されることから，無線通信リンクでの配送中の紛失が発生しうる．これは，前節の前提条件 2) によって，各移動コンピュータがすべての隣接移動コンピュータから *Creq* を受信するまでブロードキャスト送信を繰り返すことによって解決する．

Creq を受信した移動コンピュータ M_i は，現在の状態情報 $M_i.SI$ を獲得することによってローカルチェックポイント c_i を設定するとともに，*Creq* を隣接する移動コンピュータ群，すなわち，M_i の無線信号到達範囲内に存在するすべての移動コンピュータにブロードキャスト送信する．これを繰り返すことによって，前提条件 3) により，すべての移動コンピュータにおいて，ローカルチェックポイントを設定することができる．

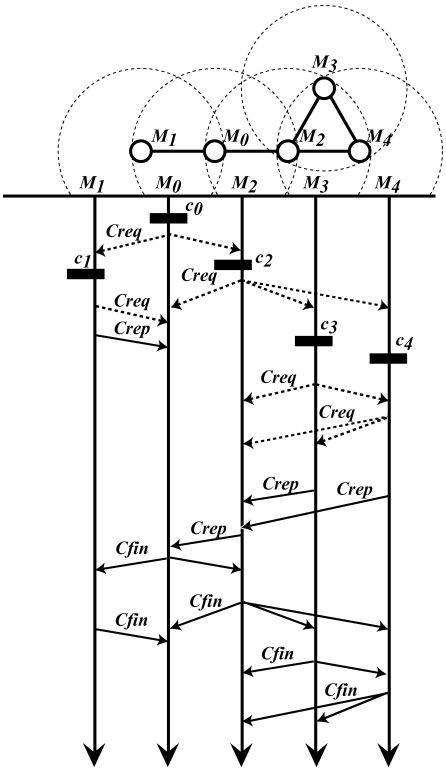

図 6 提案チェックポイント手法
Fig. 6 Overview of proposed checkpoint protocol.

チェックポイントプロトコルでは，チェックポイントにおけるコンピュータの状態情報は安定記憶に保存される．これによって，障害から回復したコンピュータが状態情報を取得し，チェックポイントからアプリケーションを再開することを可能としている．安定記憶は故障独立な複数の 2 次記憶を用いて実装するのが一般的であるが，移動コンピュータにおいては，その携帯性，移動性から実装が困難である．すべての移動コンピュータが基地局とのみ直接無線信号を交換するセル型無線ネットワークでは，基地局に安定記憶を実装して移動コンピュータの状態情報を保存するとしている．本論文では，各移動コンピュータは自身の 2 次記憶装置および隣接移動コンピュータの 2 次記憶装置に状態情報を保存する方法を用いる．各移動コン

ピュータは，少なくとも 1 つの隣接移動コンピュータを持つことから，自身の 2 次記憶に保存した状態情報が失われた場合にも，マルチホップ配送を用いて状態情報とメッセージログを取得することによってリカバリを行うことが可能である．各移動コンピュータ M_i は，ローカルチェックポイント c_i における状態情報 $M_i.SI$ を獲得し，$M_i.SI$ をピギーバックした *Creq* をブロードキャスト送信することで，追加メッセージを要することなく $M_i.SI$ を M_i の隣接移動コンピュータに配送することができる．なお，各移動コンピュータは，1 つのグローバルチェックポイントを定める *Creq* をすべての隣接移動コンピュータから受信するが，最初の *Creq* 受信時のみチェックポイント設定と *Creq* のブロードキャスト送信を行う．受信した *Creq* がすでに他の隣接移動コンピュータから受信した *Creq* と同一であることは，*Creq* のフラッディングを開始した移動コンピュータ M_0 が生成した識別子を *Creq* に付与し，これを比較することで実現できる．

Creq のフラッディング配送において，各移動コンピュータが最初に受信した *Creq* の送信元隣接移動コンピュータを親と定めると，M_0 を根とするアドホックネットワーク上のスパニングツリーが構成される．*Crep* メッセージは，このツリーに沿ってユニキャスト配送される．各移動コンピュータは，ブロードキャスト送信する *Creq* に自身の親の識別子を含める．ただし，M_0 がブロードキャスト送信する *Creq* には M_0 の識別子を含める．受信した *Creq* に自身の識別子が含まれているならば，その送信元移動コンピュータは子である．一方，他の移動コンピュータの識別子が含まれているならば，その送信元移動コンピュータは子ではない．すべての隣接移動コンピュータから *Creq* を受信した時点で子を持たない移動コンピュータは葉であり，親である隣接移動コンピュータに *Crep* をユニキャスト送信する．子であるすべての隣接移動コンピュータから *Crep* を受信した移動コンピュータは，親である移動コンピュータに *Crep* をユニキャスト送信する．これによって M_0 は子であるすべての隣接移動コンピュータから *Crep* を受信し，*Cfin* のフラッディング配送を開始する．*Cfin* のフラッディング配送は *Creq* の場合と同様に実現する．

[アドホックチェックポイントプロトコル（概略）]

1) 任意の移動コンピュータ M_0 が，M_0 の状態情報 $M_0.SI$ を獲得することでローカルチェックポイント c_0 を設定するとともに，M_0 が生成した識別子と $M_0.SI$ を含むチェックポイント設定要求メッセージ *Creq* を M_0 の無線信号到達範囲内にブロードキャスト送信する．このとき，タイマ T_0 をセットする．
2) 移動コンピュータ M_i がブロードキャスト送信した *Creq* を受信した移動コンピュータ M_j は，以下の処理を行う．
   2-1) M_i から同一の識別子を持つ *Creq* を受信していないならば，受信した *Creq* に含まれる M_i の状態情報 $M_i.SI$ を保存する．
   2-2) M_j がいずれの隣接移動コンピュータからも同一識別子を持つ *Creq* を受信していないならば，M_j の状態情報 $M_j.SI$ を獲得するとともに，$M_j.SI$ を含み，受信した *Creq* と同一の識別子を持つ *Creq* を M_j の無線信号到達範囲内にブロードキャスト送信する．このとき，タイマ T_j をセットする．
3) 移動コンピュータ M_j が隣接移動コンピュータリスト $M_j.NL$ に含まれるすべての移動コンピュータから *Creq* を受信する以前にタイマ T_j が時間切れとなったならば，M_j は，同じ識別子を持つ *Creq* を再度ブロードキャスト送信する．
4) スパニングツリーの葉である移動コンピュータ M_l は，受信した *Creq* と同一の識別子を持つチェックポイント設定応答メッセージ *Crep* を自身の親である隣接移動コンピュータにユニキャスト送信する．
5) 同一の識別子を持つ *Crep* を子であるすべての隣接移動コンピュータから受信した M_i は，受信した *Crep* と同一の識別子を持つ *Crep* を自身の親である隣接移動コンピュータにユニキャスト送信する．
6) 同一の識別子を持つ *Crep* をすべての隣接移動コンピュータから受信した M_0 は，*Crep* と同一の識別子を持つチェックポイント設定終結メッセージ *Cfin* を M_0 の無線信号到達範囲内にブロードキャスト送信する．このとき，タイマ T_0 をセットする．
7) 移動コンピュータ M_i がブロードキャスト送信した *Cfin* を受信した移動コンピュータ M_j は，いずれの隣接移動コンピュータからも同一の識別子を持つ *Cfin* を受信していないならば，受信した *Cfin* と同一の識別子を持つ *Cfin* を M_j の無線信号到達範囲内にブロードキャスト送信する．このとき，タイマ T_j をセットする．
8) 移動コンピュータ M_j が隣接移動コンピュータリスト $M_j.NL$ に含まれるすべての移動コンピュータから *Cfin* を受信する以前にタイマ T_j が時間

切れとなったならば，M_j は，同じ識別子を持つ *Cfin* を再度ブロードキャスト送信する． □

3.3 中継ログ方式

3.2 節で述べたチェックポイントプロトコルの実行と並行に送受信されたメッセージは，2.1 節で述べたように紛失メッセージや孤児メッセージとなる可能性がある．紛失メッセージは，いずれかの移動コンピュータの持つメッセージログに保存し，リカバリ時に保存されたメッセージを再配送し，送信先移動コンピュータが再受理することによってリカバリ後の移動コンピュータの状態間の矛盾を回避できる．

2 章で述べたように，従来手法として送信ログ方式と受信ログ方式がある．送信ログ方式は，メッセージの送信元移動コンピュータがメッセージログに保存する方式である．チェックポイント設定時にメッセージログに保存するメッセージの集合が確定している点が優れているが，すべての送信メッセージが保存対象となるために大きな 2 次記憶が必要となる点が欠点である．一方，受信ログ方式は，メッセージの送信先移動コンピュータがメッセージログに保存する方式である．送信先移動コンピュータは，メッセージを受信した時点でそれが紛失メッセージであるか否かを判定することができるので，紛失メッセージのみがメッセージログに保存される点で優れている．しかし，すべての紛失メッセージがメッセージログに保存されたと判定するためには，すべての移動コンピュータ間のマルチホップ配送経路に沿って，同期メッセージが配送されることが必要であり，この通信オーバヘッドを要する．

そこで，メッセージの無線マルチホップ配送経路に含まれる任意の移動コンピュータがメッセージログに保存する中継ログ方式を提案する．中継ログ方式は，各移動コンピュータがチェックポイントプロトコルの実行における *Creq* のブロードキャスト送信を行う時点でメッセージログに保存されるメッセージの集合が確定しており，紛失メッセージとなりうるメッセージのみがメッセージログに保存される．一方，孤児メッセージは，リカバリ後にアプリケーションを再開しても送信元移動コンピュータが同一のメッセージを再度送信する保証がないことから，その発生を回避しなければならない．

中継ログ方式と孤児メッセージの発生回避を行う手法を実現するために，紛失メッセージと孤児メッセージが配送経路上の中継移動コンピュータにおいて満たす性質を以下に検討する．

[性質]

1) 紛失メッセージ m_l は，その配送経路上で以下のいずれかの条件を満足する．

1-a) m_l の配送経路上にある 1 台以上の移動コンピュータ M_i において，$receive(m_l) \rightarrow c_i \rightarrow send(m_l)$ が成り立つ（**図 7**）．
ただし，$send()$ と $receive()$ は，ネットワーク層における送受信イベントであり，それぞれ■と□で記述する．また，$\rightarrow$ は，2.1 節で定義したイベント間の因果先行関係を表す．

1-b) m_l の配送経路上にある 2 台の移動コンピュータ M_i，M_j において，M_i，M_j は互いに直接通信可能であり，$send(m_l) \rightarrow c_i$ かつ $c_j \rightarrow receive(m_l)$ が成り立つ（**図 8**）．

2) 孤児メッセージ m_o は，その配送経路上で以下の条件を満足する．

2-a) m_o の配送経路上にある 2 台の移動コンピュータ M_i，M_j において，M_i と M_j は互いに直接通信可能であり，$c_i \rightarrow send(m_o)$ かつ $receive(m_o) \rightarrow c_j$ が成り立つ（**図 9**）．

□

性質 1) により，紛失メッセージとなる可能性があ

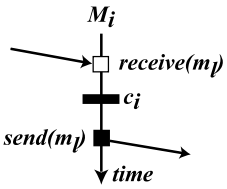

図 7 紛失メッセージの性質 1-a)
Fig. 7 Property 1-a) of lost message.

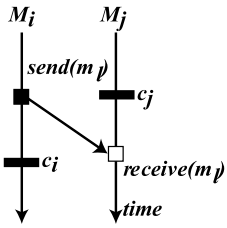

図 8 紛失メッセージの性質 1-b)
Fig. 8 Property 1-b) of lost message.

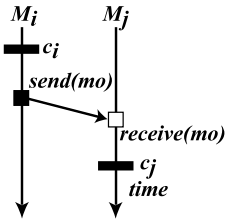

図 9 孤児メッセージの性質 2-a)
Fig. 9 Property 2-a) of orphan message.

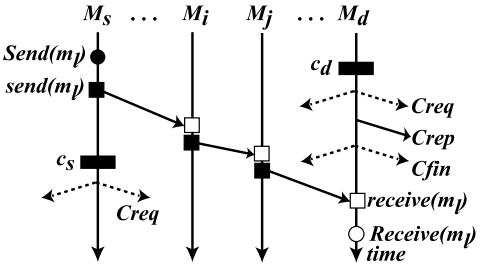

図 10 M_d による紛失メッセージの検出
Fig. 10 Lost message detection in M_d.

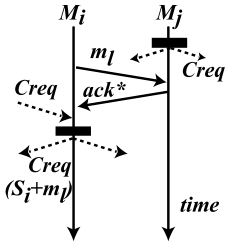

図 11 中継移動コンピュータでの紛失メッセージの検出
Fig. 11 Lost message detection in intermediate mobile computer.

るメッセージ m_l を中継移動コンピュータ，すなわち m_l の送信元でも送信先でもない移動コンピュータが検出することができる．もし，送信先移動コンピュータ M_d でのみ検出可能であるならば（たとえば，文献 7) と 15) では，M_d で m_l が紛失メッセージとなることを検出している），**図 10** に示すように，M_d が m_l を受信した時点，すなわち $Receive(m_l)$ においては，M_d がすでに $Cfin$ を隣接移動コンピュータへブロードキャスト送信済みであることが考えられる．この場合，m_l を含むメッセージログを隣接移動コンピュータの 2 次記憶に保存するためには，このメッセージログを含む追加制御メッセージをブロードキャスト送信する必要がある．

ここで，条件 1-a) を満たすメッセージ m_l については，m_l が紛失メッセージとなる可能性を持つことを移動コンピュータ M_i が $Creq$ の送信前に検出することができる．

[紛失メッセージとなる可能性の検出（1)]
移動コンピュータ M_i が中継メッセージ（M_i を送信元とも送信先ともしないメッセージ）m_l を $Creq$ のブロードキャスト送信前に受信し，$Creq$ のブロードキャスト送信時までに次ホップ移動コンピュータに転送していないならば，m_l は紛失メッセージとなる可能性のあるメッセージである．M_i は，チェックポイント c_i における状態情報 $M_i.SI$ とともに m_l を含むメッセージログを $Creq$ にピギーバックしてブロードキャスト送信する．これを受信した M_i の隣接移動コンピュータは，$M_i.SI$ とメッセージログを 2 次記憶に保存する． □

一方，条件 1-b) を満たすメッセージ m_l については，m_l が紛失メッセージとなる可能性があるメッセージであることを検出できるのは M_j であり，検出したとき M_j はすでに $Creq$ メッセージをブロードキャスト送信済みである．そこで，以下の方法により m_l が紛失メッセージとなりうるか否かの判定を M_j が行い，M_i が $Creq$ をブロードキャスト送信する以前にその判定結果を M_i に通知する．この判定結果の通知は，結果を m_l の受信確認応答メッセージにピギーバックすることによって実現される（**図 11** の $*$）．M_i において，この受信確認応答メッセージの受信が $Creq$ メッセージのブロードキャスト送信以前に行われることを保証するためには，m_l の送信からその受信確認応答メッセージの受信までの間，$Creq$ のブロードキャスト送信を禁止することが必要である．そこで，隣接移動コンピュータからメッセージを受信した移動コンピュータが受信メッセージが紛失メッセージとなりうるかを判定した結果を含む受信確認応答メッセージをただちに返送することによって，$Creq$ のブロードキャスト送信禁止によるチェックポイントプロトコル実行時間の拡大を小さくとどめる．

[紛失メッセージとなる可能性の検出（2)]
$Creq$ をブロードキャスト送信していない移動コンピュータ M_i から送信されたメッセージ m_l を移動コンピュータ M_j が $Creq$ のブロードキャスト送信後に受信したならば，m_l の受信確認応答メッセージに m_l が紛失メッセージとなる可能性があることを示す情報を付加する．M_i がこれを受信したならば，チェックポイント c_i における状態情報 $M_i.SI$ とともに m_l を含むメッセージログを $Creq$ にピギーバックしてブロードキャスト送信する．この手法を適用するために，各移動コンピュータは，メッセージを次ホップ移動コンピュータにユニキャスト転送した時点からその受信確認応答メッセージを受信するまでの間，$Creq$ をブロードキャスト送信することを禁止する． □

条件 1-a) と 1-b) はメッセージ m_l が紛失メッセージとなる必要条件であり，十分条件ではない．たとえば，**図 12** において，M_1 では $send(m) \rightarrow c_1$ であり，M_2 では $c_2 \rightarrow receive(m)$ であることから，M_2 によって紛失メッセージとなる可能性があると判定される．この結果，m は M_1 のメッセージログに保存され，その後にブロードキャスト送信される $Creq$ にピギーバックされる．しかし，M_1 では $Send(m) \rightarrow c_1$

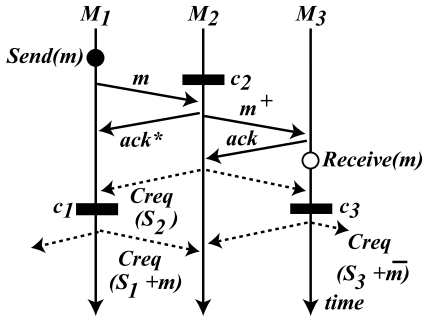

図 12 誤検出紛失メッセージの多重受理回避（誤検出の場合）
Fig. 12 Avoidance of duplicated acceptance (Consistent message case).

であり，M_3 では $Receive(m) \rightarrow c_3$ であることから，m は紛失メッセージではない．

もし，リカバリ後にグローバルチェックポイント $\mathcal{C}_{\mathcal{V}} = \{c_1, c_2, c_3\}$ からアプリケーションを再開したならば，M_1 によって再配送された m は M_3 へと配送されるが，M_3 はすでに m を受信済みの状態である．これは，中継ログ方式では紛失メッセージとなるための必要条件の充足を判定することができず，最終的に紛失メッセージとなるか否かは送信先移動コンピュータでしか判定することができないことによるものである．ここで，m が紛失メッセージとなる可能性のあるメッセージであることを検出した移動コンピュータである M_2（m をメッセージログへと保存した移動コンピュータである M_1 ではない）は，m がリカバリ時に再配送されるメッセージであることを示す情報を m に付与する（図 12 の+）．m を受信した送信先移動コンピュータ M_3 では，m の受信が $Creq$ のブロードキャスト送信以前であるならば，リカバリ時の再配送の際に m を受信してもアプリケーションが受信イベントで受理することなく破棄する必要があることを示す情報を付与して自身のメッセージログに保存する（図 12 の $\overline{m}$）．これは，M_3 における $Creq$ のブロードキャスト送信以前に行うことができるので，メッセージログを $Creq$ にピギーバックすることによってリカバリ後に必ず参照することができる．一方，m の受信が $Creq$ のブロードキャスト送信後であるならば，m は紛失メッセージであり，リカバリ後に再配送されたものをアプリケーションが受信イベントで受理する必要がある（**図 13**）．

また，**図 14** のようにマルチホップ配送経路上にある複数の中継移動コンピュータが，1 つのメッセージを紛失メッセージとなる可能性のあるメッセージであると判定することがある．たとえば，m は M_2 および M_4 において紛失メッセージとなる可能性のあるメッセージと

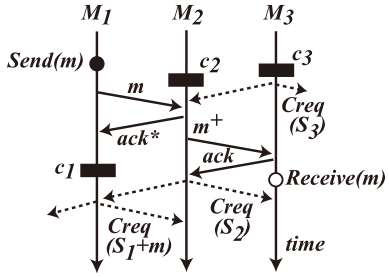

図 13 誤検出紛失メッセージの多重受理回避（誤検出でない場合）
Fig. 13 Avoidance of duplicated acceptance (Lost message case).

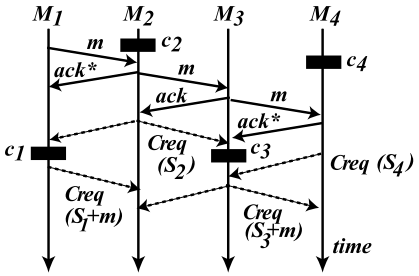

図 14 紛失メッセージの多重検出
Fig. 14 Duplicated detection of lost message.

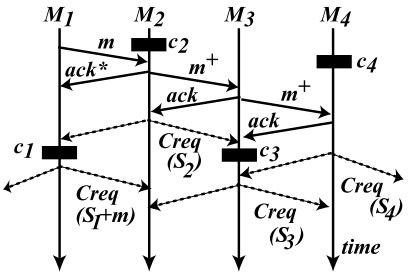

図 15 紛失メッセージの多重検出の回避
Fig. 15 Avoidance of duplicated detection of lost message.

して判定される．これは，M_1 において $send(m) \rightarrow c_1$ であり，M_2 において $c_2 \rightarrow receive(m)$ であることと，M_3 において $send(m) \rightarrow c_3$ であり，M_4 において $c_4 \rightarrow receive(m)$ であることによるものである．もし，これらの中継移動コンピュータがそれぞれのメッセージログに m を保存するならば，リカバリ時に送信先移動コンピュータ M_4 は M_1 と M_3 から再配送された 2 つの m を受信することになる．この問題を解決するために，前段で述べたリカバリ時に再配送されるメッセージであることを示す付加情報を用いる（**図 15** の +）．もし，この付加情報を含むメッセージを紛失メッセージの可能性があるメッセージであると判定しても，これをメッセージログには保存しない．これによって，このメッセージをメッセージログに保存するのは最初に紛失メッセージとなる可能性を検出した移

動コンピュータ（条件 1-a) を満たす場合），もしくはこの移動コンピュータの前ホップ移動コンピュータ（条件 1-b) を満たす場合）のみに限定され，多重にメッセージログに保存され，再配送されるという問題は解決される．

なお，条件 2) によって中継移動コンピュータによって孤児メッセージとなる可能性があることは判定できるものの，本論文で提案する手法では，これに対処することはしない．中継移動コンピュータでの判定と処理は，紛失メッセージとなる可能性のあるメッセージを含むメッセージログをチェックポイントにおける状態情報とともにチェックポイント設定要求メッセージ *Creq* にピギーバックする中継ログ方式実現のためのものである．本論文では，孤児メッセージとなる可能性のあるメッセージに対する処理は，送信先移動コンピュータでのみ行うものとする．送信先移動コンピュータは，このようなメッセージを受信した場合，自身がチェックポイントの設定を行うまでアプリケーションの受信イベントによる受理を保留することによって孤児メッセージとならないようにする．この受信イベントの生起遅延は，チェックポイントプロトコルの実行によってアプリケーションの実行を妨げるものである．これは，マルチホップ配送経路上の中継移動コンピュータにおいて条件 2) を満足する場合に発生するものであり，その発生は中継移動コンピュータのイベントスケジューリングに依存する．

3.4 チェックポイントプロトコルの開始

チェックポイントプロトコルは，ある移動コンピュータが状態情報の獲得によるローカルチェックポイントの設定とチェックポイント設定要求メッセージ *Creq* のフラッディングを開始することによって開始される．チェックポイントプロトコルはアドホックネットワークに属する任意の移動コンピュータによって開始することが可能である．開始を判断する基準として以下が考えられる．

- 最後に設定したチェックポイントプロトコルに関するイベントが生起した時刻から一定以上の時間が経過したならば，チェックポイントプロトコルを開始する．
- アドホックネットワークから退出する移動コンピュータを検出したならば，チェックポイントプロトコルを開始する．

前者の基準は，最後に設定したグローバルチェックポイントからアプリケーションの実行が十分に長い時間行われ，今後に発生するリカバリによって各移動コンピュータにおけるローカルチェックポイント以降の計算が無効化することによる損失が大きくなることによるものである．

この基準を用いる場合，複数の移動コンピュータが同時にチェックポイントプロトコルを開始することが考えられる．ここでいう同時とは，ある移動コンピュータが開始したチェックポイントプロトコルによるグローバルチェックポイントが決定する以前に，他の移動コンピュータがチェックポイントプロトコルを開始することである．この場合，各移動コンピュータが開始したチェックポイントプロトコルは，それぞれ異なるグローバルチェックポイントを定めるものではなく，単一のグローバルチェックポイントを定めるものとして扱う．これを実現するために，グローバルチェックポイントのシーケンス番号を導入する．このシーケンス番号は，アドホックネットワークに属する移動コンピュータで共通の初期値を持ち（たとえば 0），ローカルチェックポイントの設定を行うごとに 1 だけ増加されるものとする．このシーケンス番号をメッセージに含めることによって，紛失メッセージや孤児メッセージとなることやその可能性があることを判定することができる．また，チェックポイント設定要求メッセージ *Creq* に含めることによって，複数の移動コンピュータによって開始されたチェックポイントプロトコルが同一のグローバルチェックポイントを定めるものとすることができ，受信した *Creq* のブロードキャスト送信が必要か否かを判定することができる．

ただし，このような方法を用いた場合でも，ある移動コンピュータではシーケンス番号 s のグローバルチェックポイント設定を要求する *Creq* メッセージよりもシーケンス番号 $s+1$ のグローバルチェックポイント設定を要求する *Creq* メッセージを先に受信する場合が考えられる．これによって，各々のグローバルチェックポイントのために異なるメッセージログを必要とするなど，チェックポイントプロトコルを複雑にするとともに，リカバリにおいて，どのグローバルチェックポイントからアプリケーションを再開するかの決定も困難にする．そこで，各移動コンピュータでは，最新のチェックポイント設定時刻から一定時間が経過するまではチェックポイントプロトコルを開始しないものとする．この時間は，*Creq* メッセージのフラッディングに要する時間より長く設定することが好ましい．

障害により自身の 2 次記憶に保存した状態情報とメッセージログを取得できない移動コンピュータは，*Creq* メッセージにピギーバックして自身の隣接移動コンピュータに転送した状態情報とメッセージログを

取得し，リカバリを行う．そのため，チェックポイント設定時に隣接していたすべての移動コンピュータがアドホックネットワークから退出すると，この状態情報とメッセージログを取得することができなくなる．そこで，後者の基準を用いることによって解決する．

3.5 リカバリ手法

ある移動コンピュータが障害から回復した場合，この移動コンピュータでは次の状態情報取得処理がなされる．

- 自身の 2 次記憶に保存した最新のチェックポイントにおける状態情報とメッセージログが取得可能であるならば，これを取得する．
- 自身の 2 次記憶に保存した最新のチェックポイントにおける状態情報とメッセージログが取得不能であるならば，これをチェックポイント設定時に隣接していた移動コンピュータから取得する．

これによって，この移動コンピュータはリカバリに必要な情報を取得できる．

また，アドホックネットワークに属するすべての移動コンピュータに対して，リカバリを開始することを伝える必要がある．これは，リカバリ要求メッセージ $Rreq$ をフラッディングすることによって実現できる．$Rreq$ のフラッディングは，障害から回復した移動コンピュータ M_r によって開始され，$Rreq$ を受信したすべての移動コンピュータが自身の 2 次記憶から最新のチェックポイントにおける状態情報とメッセージログを取得する．M_r の状態情報とメッセージログを自身の 2 次記憶に保存している移動コンピュータは，この $Rreq$ を受信したときにこれらを M_r へマルチホップ配送する．

最後に，すべての紛失メッセージを各々の送信先移動コンピュータまで配送し，そのアプリケーションが受信イベントによって受理されることが必要である．これは，すべての移動コンピュータのメッセージログに保存されている紛失メッセージとなる可能性のあるメッセージを送信先移動コンピュータまでマルチホップ配送し，送信先移動コンピュータにおいては，このメッセージが受理せずに破棄すべきメッセージであるとして自身のメッセージログに保存されていない場合には，アプリケーションで受理することで実現される．自身のメッセージログで破棄すべきとされているメッセージは，チェックポイント取得時には中継移動コンピュータで紛失メッセージとなる可能性があるとしてメッセージログに保存されたものの，最終的に送信先移動コンピュータで受信された時点では紛失メッセージとはならなかったものである．

4. 提案プロトコル

本章では，3.3 節で提案した中継ログ方式によって紛失メッセージとなる可能性のあるメッセージを無線マルチホップ配送経路上の中継移動コンピュータが保存し，チェックポイント設定要求メッセージのフラッディングによって，このメッセージログと各移動コンピュータの状態情報を自身と隣接移動コンピュータの 2 次記憶に保存するチェックポイントプロトコルおよびリカバリプロトコルを設計する．なお，このプロトコルを実現するためには，通常のメッセージ配送においてもメッセージに必要な付加情報を追加し，適切な処理を行う必要がある．そこで，以下にメッセージ配送プロトコル，チェックポイントプロトコル，リカバリプロトコルと分けて，それぞれに必要な付加データと処理手順を示す．なお，本論文で提案する手法は，アプリケーション層とネットワーク層の中間で機能するものである．アプリケーションは，送信イベント $Send()$ と受信イベント $Receive()$ の生起を通して提案プロトコルの機能を使用し，ネットワーク層における各種メッセージの受信が提案プロトコルの各メッセージ受信イベントを生起する．このようにして，アプリケーションに対して透過的に耐故障機能を提供する．

[メッセージ配送プロトコル]

メッセージ m の経路 $||M_0, \ldots, M_n\rangle\rangle$ に沿ったマルチホップ配送に関するイベントは，以下の 4 種類である．各イベントの処理は，各々に記述された条件を満足することによって開始可能となる．各移動コンピュータは，開始可能となった順にイベントの処理をスケジュールするものとする．

なお，各移動コンピュータ M_i は，以下の情報を保持する．

- $M_i.cseq$：M_i が設定した最新チェックポイントのシーケンス番号
- $M_i.ML^+$：紛失メッセージ候補の集合である肯定メッセージログ
- $M_i.ML^-$：他の移動コンピュータのメッセージログに記録されている紛失メッセージではないメッセージの集合である否定メッセージログ

無線マルチホップ配送の各転送ホップにおけるメッセージ m には，以下の情報を付加する．

- $m.logged$：無線マルチホップ配送中に経路上のいずれかの移動コンピュータ M_i のメッセージログ $M_i.ML^+$ に m が記録されたか否かを示す真偽値
- $m.src\_cseq$：無線マルチホップ配送の送信元移動コンピュータ M_0 における m の送信イベント時

の最新チェックポイントシーケンス番号 $M_0.cseq$ の値

- $m.sndr\_cseq$：各転送ホップの転送元移動コンピュータ M_i における m の送信イベント時の最新チェックポイントシーケンス番号 $M_i.cseq$ の値

また，各転送ホップにおけるメッセージ m に対する受信確認応答メッセージ ack_m には，以下の情報を付加する．

- $ack_m.logging$：転送元移動コンピュータ M_i のメッセージログ $M_i.ML^+$ への m の記録が必要か否かを示す真偽値

送信元移動コンピュータ M_0 における送信イベント $src\_send(m)$

[条件] アプリケーションの送信イベント $Send(m)$ が生起する．

[処理]

1. 以下のメッセージ m を作成する．
   - $m.logged := false$ とする．
   - $m.src\_cseq := M_0.cseq$ とする．
   - $m.sndr\_cseq := M_0.cseq$ とする．
2. m を転送先移動コンピュータ M_1 に送信する．
3. ack_m を M_1 から受信する．
4. $ack_m.logging = true$ であるならば，m を $M_0.ML^+$ に記録する．

中継移動コンピュータ M_i における受信イベント $int\_receive(m)$

[条件] 転送元移動コンピュータ M_{i-1} から m を受信する．

[処理]

1. 以下の受信確認応答メッセージ ack_m を作成する．
   - $m.logged = false$ かつ $m.sndr\_cseq < M_i.cseq$ であるならば，$ack_m.logging := true$，それ以外の場合は $ack_m.logging := false$ とする．
2. ack_m を M_{i-1} に送信する．

中継移動コンピュータ M_i における送信イベント $int\_send(m)$

[条件] M_i において，受信イベント $int\_receive(m)$ の処理が終了する．

[処理]

1. $int\_receive(m)$ で M_{i-1} から受信した m について以下を変更する．
   - $m.sndr\_cseq < M_i.cseq$ であるならば $m.logged := true$ とする．
   - $m.sndr\_cseq := M_i.cseq$ とする．
2. m を転送先移動コンピュータ M_{i+1} に送信する．
3. ack_m を M_{i+1} から受信する．
4. $ack_m.logging = true$ であるならば，m を $M_i.ML^+$ に記録する．

送信先移動コンピュータ M_n における受信イベント $dest\_receive(m)$

[条件] 転送元移動コンピュータ M_{n-1} から m を受信する．

[処理]

1. 以下の受信確認応答メッセージ ack_m を作成する．
   - $m.logged = false$ かつ $m.sndr\_cseq < M_n.cseq$ であるならば $ack_m.logging := true$，それ以外の場合は $ack_m.logging := false$ とする．
2. ack_m を M_{n-1} に送信する．
3. $m \in M_n.ML^-$ であるならば，m を $M_n.ML^-$ から除去する．
4. $m.logged = true$ かつ $m.src\_cseq = M_n.cseq$ であるならば，m を $M_n.ML^-$ に記録する．

$dest\_receive(m)$ を終えた M_n は，m が処理 3 において $M_n.ML^-$ から除去されたものでなければ，$m.src\_cseq = M_n.cseq$ を満足することを条件にアプリケーションの受信イベント $Receive(m)$ による m の受理を可とする．$m.src\_cseq > M_n.cseq$ であるならば，$m.src\_cseq = M_n.cseq$ となるまで $Receive(m)$ による m の受理を保留する．

[チェックポイントプロトコル]

チェックポイントプロトコルにおける，アドホックネットワークを構成する移動コンピュータで生起するイベントは，以下の 7 種類である．各イベントの処理は，各々に記述された条件を満足することによって開始可能となる．各移動コンピュータは，開始可能となった順にイベントの処理をスケジュールするものとする．

なお，各移動コンピュータ M は，以下の情報を保持する．

- $M.cseq$：M が設定した最新チェックポイントのシーケンス番号
- $M.ML^+$：紛失メッセージ候補の集合である肯定メッセージログ
- $M.ML^-$：他の移動コンピュータのメッセージログに記録されている紛失メッセージではないメッセージの集合である否定メッセージログ
- $M.SI$：チェックポイント設定時に獲得した状態情報
- $M.ckpt$：チェックポイントプロトコルが進行中であるか否かを示す真偽値

チェックポイント設定要求メッセージ $Creq$ には，以

下の情報を付加する．

- $Creq.cseq$：$Creq$ の送信元移動コンピュータ M における $Creq$ 送信イベント時の最新チェックポイントシーケンス番号 $M.cseq$ の値
- $Creq.SI$：$Creq$ の送信元移動コンピュータ M がチェックポイント設定時に獲得した状態情報 $M.SI$
- $Creq.ML^+$：$Creq$ の送信元移動コンピュータ M のチェックポイント設定時における肯定メッセージログ $M.ML^+$
- $Creq.ML^-$：$Creq$ の送信元移動コンピュータ M のチェックポイント設定時における否定メッセージログ $M.ML^-$

チェックポイント設定応答メッセージ $Crep$ には，以下の情報を付加する．

- $Crep.cseq$：$Crep$ の送信元移動コンピュータ M における $Crep$ 送信イベント時の最新チェックポイントシーケンス番号 $M.cseq$ の値

チェックポイント設定終結メッセージ $Cfin$ には，以下の情報を付加する．

- $Cfin.cseq$：$Cfin$ の送信元移動コンピュータ M における $Cfin$ 送信イベント時の最新チェックポイントシーケンス番号 $M.cseq$ の値

移動コンピュータ M_p におけるチェックポイント開始イベント *invoke_checkpoint*()

[条件] 以下のいずれかを満足する．

- 最新のチェックポイントを設定してから最大チェックポイント間隔 $M_p.\Delta$ が経過する．
- 最新のチェックポイントを設定してから最小チェックポイント間隔 $M_p.\delta$ が経過し，隣接移動コンピュータのアドホックネットワークからの退出を検出する．

[処理]

1. M_p の状態情報 $M_p.SI$ を獲得する．
2. $int\_receive(m)$ の処理が終了し，$int\_send(m)$ の処理が終了していない $m.logged = false$ であるすべてのメッセージ m を $m.logged := true$ として $M_p.ML^+$ に記録する．
3. $M_p.SI$，$M_p.ML^+$，$M_p.ML^-$ を保存する．
4. $M_p.cseq := M_p.cseq + 1$ とする．

移動コンピュータ M_p におけるチェックポイント設定要求メッセージ受信イベント *creq_receive*(*Creq*)

[条件] 隣接移動コンピュータ M_q から $Creq$ を受信する．

[処理]

1. $Creq.SI$，$Creq.ML^+$，$Creq.ML^-$ を保存する．
2. $M_p.cseq < Creq.cseq$ であるならば，以下の処理を行う．
   2-1. M_p の状態情報 $M_p.SI$ を獲得する．
   2-2. $int\_receive(m)$ の処理が終了し，$int\_send(m)$ の処理が終了していない $m.logged = false$ であるすべてのメッセージ m を $m.logged := true$ として $M_p.ML^+$ に記録する．
   2-3. $M_p.SI$，$M_p.ML^+$，$M_p.ML^-$ を保存する．
   2-4. $M_p.cseq := Creq.cseq$ とする．

移動コンピュータ M_p におけるチェックポイント設定要求メッセージ送信イベント *creq_send*(*Creq*)

[条件] 以下のいずれかを満足する．

- M_p において，$invoke\_checkpoint()$ が終了する．
- $M_p.cseq < Creq.cseq$ を満足する $Creq$ の受信イベント $creq\_receive(Creq)$ が終了する．

[処理]

1. 以下のチェックポイント設定要求メッセージ $Creq$ を作成する．
   - $Creq.cseq := M_p.cseq$ とする．
   - $Creq.SI := M_p.SI$ とする．
   - $Creq.ML^+ := M_p.ML^+$ とする．
   - $Creq.ML^- := M_p.ML^-$ とする．
2. $Creq$ をすべての隣接移動コンピュータにブロードキャスト送信する．
3. $M_p.ML^+ := \emptyset$ とする．
4. $M_p.ML^- := \emptyset$ とする．
5. $M_p.ckpt := true$ とする．

移動コンピュータ M_p におけるチェックポイント設定応答メッセージ送信イベント *crep_send*(*Crep*)

[条件] 以下の両方の条件を満足する．

- すべての隣接移動コンピュータから $Creq$ を受信済みである．
- 子であるすべての隣接移動コンピュータから受信した $Crep$ について $crep\_receive(Crep)$ が終了する．

[処理]

1. 以下のチェックポイント設定応答メッセージ $Crep$ を作成する．
   - $Crep.cseq := M_p.cseq$ とする．
2. $Crep$ を親である隣接移動コンピュータへ送信する．

移動コンピュータ M_p におけるチェックポイント設定応答メッセージ受信イベント *crep_receive*(*Crep*)

[条件] 隣接移動コンピュータ M_q から $Crep$ を受信する．

[処理] 処理内容なし

移動コンピュータ M_p におけるチェックポイント設定終結メッセージ送信イベント *cfin_send*(*Cfin*)

[条件] 以下のいずれかの条件を満足する．

- M_p において *invoke_checkpoint*() が終了し，すべての隣接移動コンピュータから受信した *Crep* について *crep_receive*(*Crep*) が終了する．
- *cfin_receive*(*Cfin*) が終了し，$M_p.ckpt = true$ である．

[処理]

1. 以下のチェックポイント設定終結メッセージ *Cfin* を作成する．
   - $Cfin.cseq := M_p.cseq$ とする．
2. *Cfin* をすべての隣接移動コンピュータにブロードキャスト送信する．
3. $M_p.ckpt := false$ とする．

移動コンピュータ M_p におけるチェックポイント設定終結メッセージ受信イベント *cfin_receive*(*Cfin*)

[条件] 隣接移動コンピュータ M_q から *Cfin* を受信する．

[処理] 処理内容なし

[リカバリプロトコル]

リカバリプロトコルにおける，アドホックネットワークを構成する移動コンピュータで生起するイベントは，以下の 4 種類である．各イベントの処理は，各々に記述された条件を満足することによって開始可能となる．各移動コンピュータは，開始可能となった順にイベントの処理をスケジュールするものとする．

なお，各移動コンピュータ M は，以下の情報を保持する．

- $M.cseq$：M が設定した最新チェックポイントのシーケンス番号
- $M.rcvry$：リカバリが進行中であるか否かを示す真偽値

リカバリ要求メッセージ *Rreq* には，以下の情報を付加する．

- $Rreq.cseq$：リカバリすべきチェックポイントのシーケンス番号
- $Rreq.src$：送信元移動コンピュータの識別子
- $Rreq.sreq$：送信元移動コンピュータが状態情報，メッセージログの転送を要求するか否かを示す真偽値

また，リカバリ応答メッセージ *Rrep* には，以下の情報を付加する．

- $Rrep.SI$：*Rrep* の送信先移動コンピュータ M がリカバリに用いる状態情報 $M.SI$
- $Rrep.ML^+$：*Rrep* の送信先移動コンピュータ M がリカバリに用いる肯定メッセージログ $M.ML^+$
- $Rrep.ML^-$：*Rrep* の送信先移動コンピュータ M がリカバリに用いる否定メッセージログ $M.ML^-$

障害回復移動コンピュータ M_r におけるリカバリ開始イベント *invoke_recovery*()

[条件] M_r が障害から回復する．

[処理]

1. 以下のリカバリ要求メッセージ *Rreq* を作成する．
   - $Rreq.cseq := M_p.cseq$ とする．
   - $Rreq.src := M_r$ とする．
   - M_r が最新チェックポイントにおける自身の状態情報，肯定メッセージログ，否定メッセージログを 2 次記憶から取得可能である場合は $Rreq.sreq := false$ とする．取得不能である場合は $Rreq.sreq := true$ とする．
2. *Rreq* をすべての隣接移動コンピュータにブロードキャスト送信する．
3. $M_r.rcvry := true$ とする．

移動コンピュータ M_p におけるリカバリ要求メッセージ受信イベント *rreq_receive*(*Rreq*)

[条件] 隣接移動コンピュータ M_q から *Rreq* を受信する．

[処理]

1. $M_p.rcvry = false$ ならば，$M_p.rcvry := true$ として以下の処理を行う．
   - 1-1. $Rreq.sreq = true$ であり，$Rreq.src$ の状態情報 $Rreq.src.SI$，肯定メッセージログ $Rreq.src.ML^+$，否定メッセージログ $Rreq.src.ML^-$ を保存している場合，以下の処理を行う．
     - 1-1-1. 以下のリカバリ応答メッセージ *Rrep* を作成する．
       - $Rrep.SI := Rreq.src.SI$ とする．
       - $Rrep.ML^+ := Rreq.src.ML^+$ とする．
       - $Rrep.ML^- := Rreq.src.ML^-$ とする．
     - 1-1-2. *Rrep* を $Rreq.src$ へマルチホップ送信する．
   - 1-2. 最新チェックポイントにおける自身の状態情報 $M_p.SI$，肯定メッセージログ $M_p.ML^+$，否定メッセージログ $M_p.ML^-$ を 2 次記憶から取得する．
   - 1-3. $M_p.ML^+$ に含まれるすべてのメッセージ m を *int_send*(m) によって再配送する．
   - 1-4. $M_p.ML^+ := \emptyset$ とし，$M_p.SI$ からアプリ

ケーションを再開する．

2. すべての隣接移動コンピュータから $Rreq$ を受信したならば，$M_p.rcvry := false$ とする．

移動コンピュータ M_p におけるリカバリ要求メッセージ送信イベント $rreq\_send(Rreq)$

[条件] $rreq\_receive(Rreq')$ を終了する．

[処理]

1. 以下のリカバリ要求メッセージ $Rreq$ を作成する．
   - $Rreq.cseq := Rreq'.cseq$ とする．
   - $Rreq.sreq := Rreq'.sreq$ とする．
2. $Rreq$ をすべての隣接移動コンピュータにブロードキャスト送信する．
3. $M_p.rcvry := true$ とする．

障害回復移動コンピュータ M_r におけるリカバリ応答メッセージ受信イベント $rrep\_receive(Rrep)$

[条件] マルチホップ配送された $Rrep$ を受信する．

[処理]

1. 最新チェックポイントにおける自身の状態情報，肯定メッセージログ，否定メッセージログを $M_r.SI := Rrep.SI$，$M_r.ML^+ := Rrep.ML^+$，$M_r.ML^- := Rrep.ML^-$ により $Rrep$ から取得する．
2. $M_r.ML^+$ に含まれるすべてのメッセージ m を $int\_send(m)$ によって再配送する．
3. $M_r.ML^+ := \emptyset$ とし，$M_r.SI$ からアプリケーションを再開する．

なお，再配送されたメッセージ m について，送信先移動コンピュータにおいて再受理が必要であるかを判定しなければならない．これは，$dest\_receive(m)$ の処理において実現されている．

5. 評　　価

本論文で提案する中継ログ方式と従来手法である送信ログ方式および受信ログ方式をアドホックネットワークに適用した場合の通信オーバヘッドについて評価する．なお，以下で示すシミュレーション実験にはGloMoSim [30] を用いた．

5.1 メッセージ数の比較

まず，プロトコルの構成からメッセージ数を比較する．中継ログ方式と送信ログ方式は，メッセージログが $Creq$ 送信時点で決定している方式である．したがって，Koo らのプロトコルを基礎とすることで状態情報とメッセージログを $Creq$ にピギーバックすることが可能であり，$Creq$ のフラッディング配送，$Crep$ のスパニングツリーに沿ったユニキャスト配送，$Cfin$ のフラッディング配送が交換されるメッセージである．一方，受信ログ方式では，メッセージログが $Creq$ 送信時点では決定していないため，状態情報のみを $Creq$ にピギーバックできる．すべての移動コンピュータ対を接続するマルチホップ配送経路に配送途中の紛失メッセージがないことを確認するために，同期メッセージをこの経路に沿って配送する必要がある．さらに，各移動コンピュータでは，他のすべての移動コンピュータからこの同期メッセージを受信した時点でメッセージログが決定され，これを隣接移動コンピュータへ配送するためにメッセージログをピギーバックしたメッセージをブロードキャストする．このメッセージ数は，フラッディング配送と同数となる．以上により，メッセージ数の比較では，送信ログ方式 = 中継ログ方式 < 受信ログ方式となる．

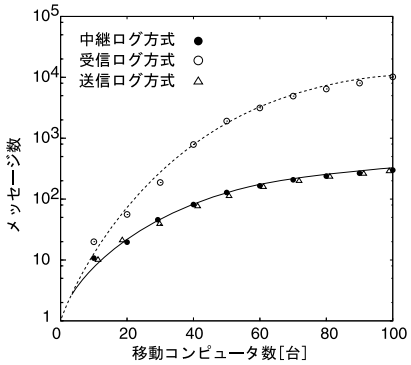

図 16 メッセージ数の比較
Fig. 16 Number of control messages.

次に，シミュレーション実験によりメッセージ数を比較する．各移動コンピュータの無線信号到達距離を 100 m とし，500 m × 500 m のシミュレーション領域に 10 台から 100 台まで 10 台おきの移動コンピュータをランダムに配置する．なお，チェックポイントプロトコル動作中の移動コンピュータ位置は固定であり，障害は発生しないとした．シミュレーション実験結果を**図 16** に示す．

ここで，x 軸は移動コンピュータ数を，y 軸は 1 つのグローバルチェックポイントを定めるためにアドホックネットワーク全体で交換された総メッセージ数を表している．前段の考察のとおり，中継ログ方式と送信ログ方式はほぼ同数のメッセージが交換され，受信ログ方式よりも少ない数である．本実験の範囲では，中継ログ方式のメッセージ数は受信ログ方式に比べて最大 97.1%，最小 75.0%，平均 92.0%の削減となっている．

5.2 メッセージ量の比較

前節同様，まず，プロトコルの構成からメッセージ量を比較する．アドホックネットワークでは，障害か

ら回復した移動コンピュータが自身の 2 次記憶に保存した状態情報とメッセージログを取得できるとは限らないことから，本論文では，チェックポイント設定時における隣接移動コンピュータの 2 次記憶にもこれらを格納する手法を提案している．3 つのログ方式すべてにこの手法を適用した場合，いずれの方法においても各移動コンピュータは，状態情報とメッセージログをいずれかのメッセージにピギーバックして 1 度ブロードキャスト送信することになる．そこで，これらのメッセージログの大きさについて検討する．送信ログ方式では，各移動コンピュータは，前回のローカルチェックポイント設定以降に送信したすべてのメッセージをメッセージログに保存するため，アドホックネットワーク全体では，前回のグローバルチェックポイント以降に送信されたすべてのメッセージが 1 度ずつメッセージログに保存される．受信ログ方式では，各移動コンピュータは，今回のグローバルチェックポイント設定時の紛失メッセージのみをそれぞれ 1 度だけメッセージログに保存する．したがって，メッセージログの大きさの比較では，受信ログ方式 < 送信ログ方式である．一方，中継ログ方式は，各移動コンピュータが今回のローカルチェックポイント設定時に紛失メッセージとなる可能性があると判断された中継メッセージをメッセージログに保存する．このとき，3 章で述べた誤検出が発生しなければ，メッセージログの大きさは受信ログの場合に等しいが，これらが発生した場合には，その分だけ大きくなる．メッセージ量には，メッセージログの大きさ，状態情報の大きさ，メッセージ数が影響を与える．

次に，シミュレーション実験によりメッセージ量を比較する．前節と同様，各移動コンピュータの無線信号到達距離を 100 m とし，500 m × 500 m のシミュレーション領域に 10 台から 100 台まで 10 台おきの移動コンピュータをランダムに配置する．各移動コンピュータは，512 バイトのデータメッセージを 1.5 秒間隔でランダムに選択した移動コンピュータに送信する．チェックポイント取得間隔を 60 秒とし，各移動コンピュータが取得する状態情報のサイズを 1,024 バイト，2,048 バイト，4,096 バイト，8,192 バイトとする．また，提案プロトコルの交換する制御メッセージサイズは 12 バイトとし，すべてのメッセージには 50 バイトのヘッダが付与される．なお，前節と同様，チェックポイントプロトコル動作中の移動コンピュータ位置は固定であり，障害は発生しないとした．

シミュレーション実験結果として，各移動コンピュータが送信したメッセージ量の平均値を**図 17** に示す．

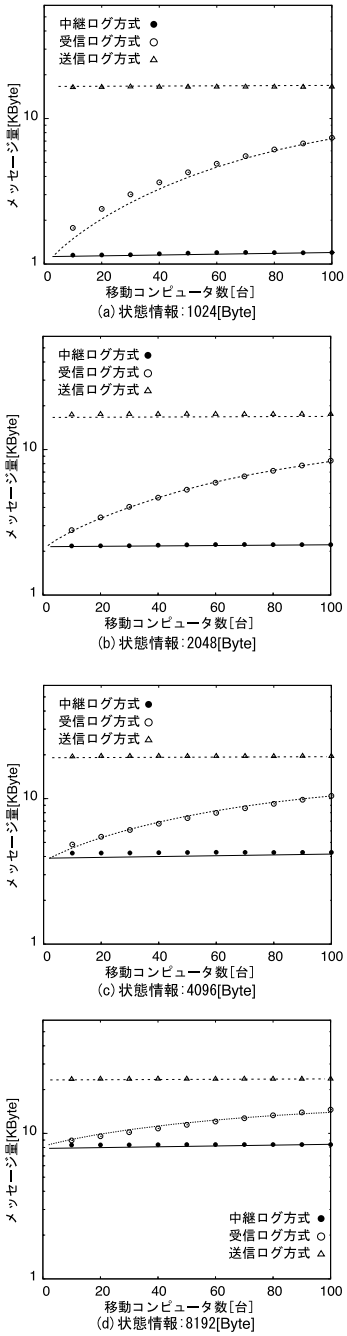

図 17 メッセージ量の比較
Fig. 17 Amount of control messages.

総量では，中継ログ方式が最も少なく，受信ログ方式，送信ログ方式の順となることが分かる．各移動コンピュータは，いずれの方式においても状態情報を 1 度だけ隣接移動コンピュータにブロードキャスト送信することから，それぞれの方式におけるメッセージ量の違いは，メッセージログ量と制御メッセージ量による

ものである．中継ログ方式と送信ログ方式は制御メッセージ数がほぼ同じであることから，この差はメッセージログ量によるものである．また，中継ログ方式と送信ログ方式では，移動コンピュータが送信する制御メッセージ量は移動コンピュータ数に依存しないのに対して，受信ログ方式では，すべての紛失メッセージを検出するために交換する同期制御メッセージ量が移動コンピュータ数の増加にともなって増加している．

ここで，メッセージログ量の比較結果を**図 18** に示す．すべての送信メッセージをメッセージログに格納する送信ログ方式に対して，紛失メッセージおよびその候補のみをメッセージログに格納する受信ログ方式を中継ログ方式が大幅に下回っていることが分かる．紛失メッセージの誤検出分だけ中継ログ方式が受信ログ方式よりも多くメッセージログに格納しており，その割合は平均で 30.5%である．誤検出が頻繁に発生すると中継ログ方式のメッセージ量が受信ログ方式よりも大きくなることが考えられるが，図 17 の実験結果では移動コンピュータ数が少ない場合にほぼ同等になるものの，それ以外では一貫して中継ログ方式のメッセージ量は最小となった．

5.3 プロトコル動作時間の比較

チェックポイントプロトコルの動作時間は，送信ログ方式と中継ログ方式では *Creq* と *Cfin* のフラッディング配送と *Crep* のユニキャスト配送の時間の合計であるのに対して，受信ログ方式では *Creq* のフラッディング配送と紛失メッセージ検出のための同期制御メッセージの交換，メッセージログを含んだ制御メッセージのブロードキャストの時間の合計である．ただし，*Creq* のブロードキャスト送信は，データメッセージの *ack* を受信待ちしている移動コンピュータでは一時停止されるため，この時間だけ中継ログ方式の方が長くなることが考えられる．また，アドホックネットワークの規模に強く依存し，移動性の影響は小さいと考えられることから，シミュレーション実験は，5.2 節と同じ環境で行った．実験結果を**図 19** に示す．中継ログ方式と送信ログ方式は受信ログ方式よりも要する時間が短く，平均 83.0%の短縮となっている．中継ログ方式と送信ログ方式の差は平均 3.24%であり，*Creq* のブロードキャスト送信待ちの影響は小さいといえる．

5.4 移動性の影響

提案するチェックポイントプロトコルの動作時間は，5.3 節のシミュレーション実験結果に示すように短いものである．そのため，移動コンピュータの移動性による性能の変化は大きくないと考えられる．一方，リカバリプロトコルの性能は，移動性によってその性質

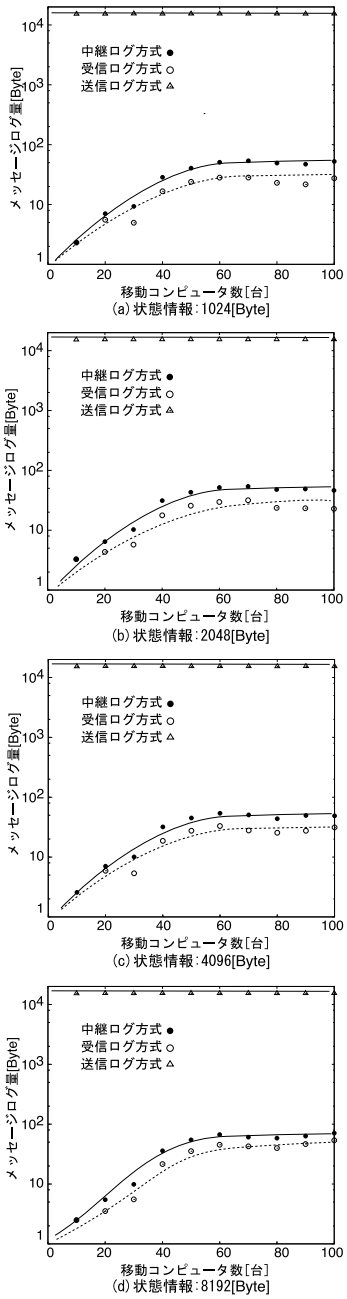

図 18 メッセージログ量の比較
Fig. 18 Amount of logged messages.

が異なることが考えられる．各移動コンピュータは，チェックポイント設定時に隣接移動コンピュータに状態情報とメッセージログを格納する．障害から回復した移動コンピュータは，この状態情報とメッセージログを取得してリカバリを行うが，これらを格納した移動コンピュータがリカバリ時点で隣接しているとは限らず，マルチホップ配送を必要とする．このマルチホッ

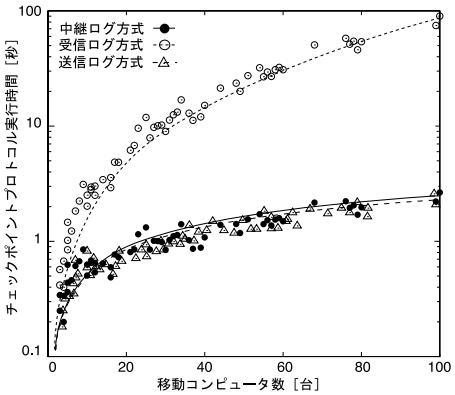

図 19 チェックポイントプロトコルの実行時間
Fig. 19 Execution time of checkpoint protocol.

プ配送ホップ数は，移動性とチェックポイント設定からリカバリまでの時間によって異なる．そこで，リカバリ時間をシミュレーション実験によって評価する．

前節までのシミュレーション実験と同様，各移動コンピュータの無線信号到達距離を 100 m とし，500 m × 500 m のシミュレーション領域に 10 台から 100 台まで 10 台おきの移動コンピュータをランダムに配置する．各移動コンピュータの移動はランダムウエイポイントに基づくものとし，その平均移動速度を毎秒 1 m から 8 m まで 1 m おきに与えて実験を行う．チェックポイント設定時刻からの経過時間に対する各移動コンピュータにチェックポイント設定時に隣接していた移動コンピュータのうち最も近いものまでのホップ数を**図 20** に示す．ただし，移動コンピュータ数は 60 台，80 台，100 台であるとする．速度が大きいほど，また時間が経過するほど，ホップ数が大きくなる．したがって，チェックポイント設定時の隣接移動コンピュータに格納した状態情報とメッセージログを取得するためにマルチホップ配送を使う必要性が増加することが分かる．

また，経過時間に対するリカバリプロトコル動作時間を各移動速度について測定した結果を**図 21** に示す．チェックポイント設定時刻からの時間経過が小さい段階では，状態情報が比較的各移動コンピュータの近隣にあるため，リカバリプロトコル動作時間は *Rreq* の配送時間に支配されている．経過時間が大きくなると状態情報のマルチホップ配送に要する時間の影響でリカバリ時間が拡大すると思われたが，結果は時間が短縮した後，少しずつ拡大するというものであった．これは，実験領域が無線信号到達距離に対して比較的小さいこと，ランダムウエイポイントモデルに基づいて移動した結果，移動コンピュータの分布が領域の中央

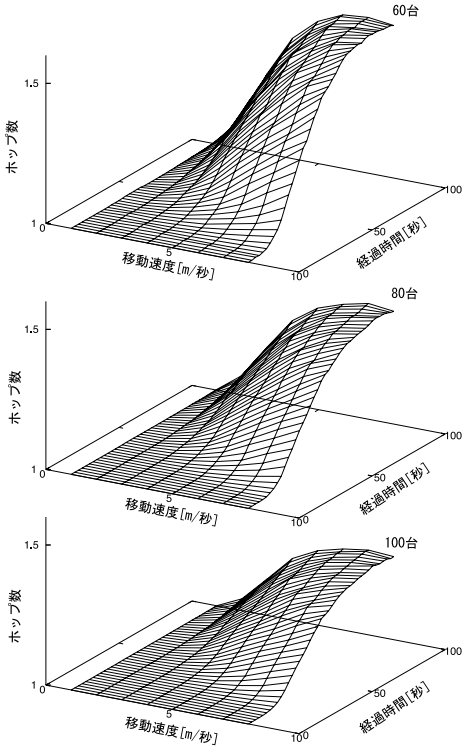

図 20 状態情報を持つ移動コンピュータまでのホップ数
Fig. 20 Hop count for state information transmission.

で高くなったことが考えられる．ただし，50 秒以降の漸増傾向はマルチホップ配送による影響であると考えられる．

5.5 孤児メッセージ数の評価

中継ログ方式では，チェックポイントプロトコル実行中の送信先コンピュータにおける *Receive*(m) イベントの生起を遅延させることで孤児メッセージの発生を回避している．この受信イベント生起遅延が多く発生すると，アプリケーションメッセージの配送遅延が大きくなるという問題がある．そこで，この受信イベント生起遅延を必要とするメッセージの割合をシミュレーション実験評価する．実験環境は 5.2 節に示したものであるが，この割合は各移動コンピュータでのチェックポイント設定時間に依存すると考えられているため，状態情報の大きさを変化させて実験を行った．結果を**図 22** に示す．孤児メッセージ発生回避のための受信イベント生起遅延発生率は，状態情報の大きさが大きくなるとともに拡大しているが，0.5%以下にとどまっている．そのため，アプリケーション実行への影響は小さいと考えられる．

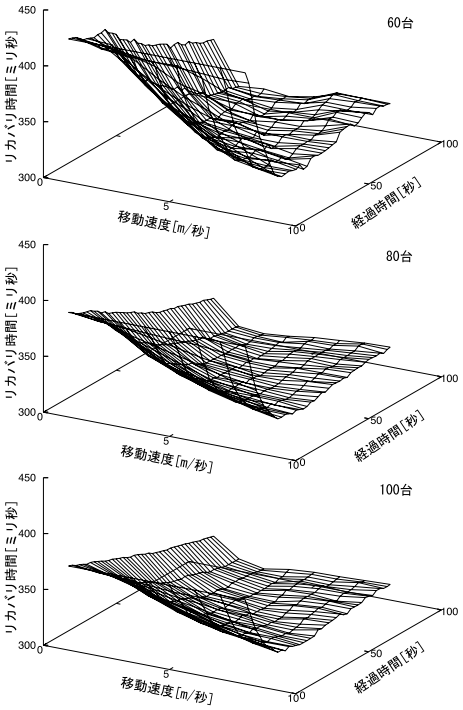

図 21 リカバリプロトコルの動作時間
Fig. 21 Execution time of recovery protocol.

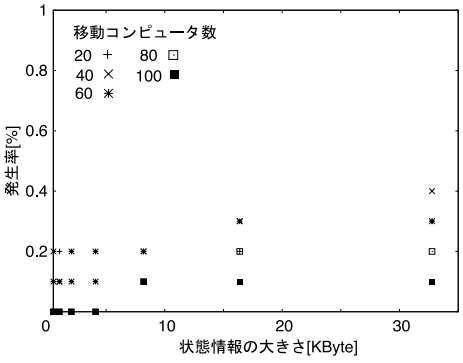

図 22 孤児メッセージ候補の発生率
Fig. 22 Ratio of orphan messages.

6. ま と め

本論文では，アドホックネットワークのためのチェックポイントリカバリ手法を提案し，その実現プロトコルを設計した．無線マルチホップ配送によってメッセージが配送されるアドホックネットワークにおいて，紛失メッセージを従来提案されている送信ログ方式，受信ログ方式を用いて保存する場合の問題点を示し，中継ログ方式を提案した．中継ログ方式を用いることにより，紛失メッセージとなる可能性のあるメッセージを中継移動コンピュータのチェックポイントにおける状態情報とともに，この中継移動コンピュータとその隣接移動コンピュータの 2 次記憶に保存することができる．これらの配送はフラッディングされるチェックポイント設定要求メッセージへのピギーバックによって実現できることから，追加制御メッセージが不要である．また，シミュレーション実験結果からほとんどの場合に配送メッセージ量も従来手法より小さく，通信オーバヘッドの小さなチェックポイントプロトコルとなっている．

本論文で提案したチェックポイントプロトコルは，その処理中にアドホックネットワークからの退出がないことを前提としている．より大規模なアドホックネットワークへの適用を考慮する場合には，チェックポイントプロトコル実行中のアドホックネットワークトポロジの変化に対してより高い耐性を持つ手法を考案することが必要である．

参 考 文 献

1) Acharya, A. and Badrinath, B.R.: Checkpointing Distributed Applications on Mobile Computers, *Proc. 3rd International Conference on Parallel and Distributed Information Systems*, pp.73–80 (1994).
2) Ahn, J., Min, S.G. and Hwang, C.S.: A Causal Message Logging Protocol for Mobile Nodes in Mobile Computing Systems, *Future Generation Computer Systems*, Vol.20, No.4, pp.663–686 (2004).
3) Alvisi, L., Hoppe, B. and Marzullo, K.: Nonblocking and Orphan-Free Message Logging Protocols, *Proc. 23rd International Symposium on Fault-Tolerant Computing*, pp.145–154 (1993).
4) Alvisi, L. and Marzullo, K.: Message logging: Pessimistic, Optimistic, Causal, and Optimal, *IEEE Trans. Softw. Eng.*, Vol.24, No.2, pp.149–159 (1998).
5) Callaway, E.H.: *Wireless Sensor Networks*, Auerbach Publications (2003).
6) Cao, G. and Singhal, M.: Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.12, No.2, pp.157–172 (2001).
7) Chandy, K.M. and Lamport, L.: Distributed Snapshots: Determining Global States of Distributed Systems, *ACM Trans. Computer Systems*, Vol.3, pp.63–75 (1985).
8) Corson, M.S. and Ephremides, A.: A Distributed Routing Algorithm for Mobile Wire-

less Networks, *ACM Journal of Wireless Networks*, Vol.1, No.1, pp.61–81 (1995).
9) Elnozahy, E.N., Alvisi, L., Wang, Y.M. and Johnson, D.: A Survey of Rollback-Recovery Protocols in Message-Passing Systems, *ACM Computing Surveys*, Vol.34, No.3, pp.375–408 (2002).
10) ETSI Functional Specifications: Radio Equipment and Systems (RES); HIPERLAN (1995).
11) Gendelman, E., Bic, L.F. and Dillencourt, M.B.: An Efficient Checkpointing Algorithm for Distributed Systems Implementing Reliable Communication Channels, *Proc. 18th International Symposium on Reliable Distributed Systems*, pp.290–291 (1999).
12) Johnson, D.B. and Zwaenepoel, W.: Sender-Based Message Logging, *Proc. 17th International Symposium on Fault-Tolerant Computing*, pp.14–19 (1987).
13) Johnson, D.B. and Zwaenepoel, W.: Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing, *Proc. 7th Annual ACM Symposium on Principles of Distributed Computing*, pp.171–181 (1988).
14) Kim, J.L. and Park, T.: An Efficient Protocol for Checkpointing Recovery in Distributed Systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.4, No.8, pp.955–960 (1993).
15) Koo, R. and Toueg, S.: Checkpointing and Rollback–Recovery for Distributed Systems, *IEEE Trans. Softw. Eng.*, Vol.SE-13, No.1, pp.23–31 (1987).
16) Lampson, B.W.: *Atomic Trans.*, Vol.105, Springer Verlag (1981).
17) Lesser, V., Ortiz, C. and Tambe, M.: *Distributed Sensor Networks*, Kluwer Academic Publications (2003).
18) Neves, N. and Fuchs, W.K.: Adaptive Recovery for Mobile Environments, *Comm. ACM*, Vol.40, No.1, pp.68–74 (1997).
19) Park, T., Woo, N. and Yeom, H.Y.: An Efficient Recovery Scheme for Fault-tolerant Mobile Computing Systems, *Future Generation Computer Systems*, Vol.19, No.1, pp.37–53 (2003).
20) Park, T., Woo, N. and Yeom, Y.: Efficient Recovery Information Management Schemes for the Fault Tolerant Mobile Computing Systems, *Proc. 20th International Symposium on Reliable Distributed Systems*, pp.202–205 (2001).
21) Park, T. and Yeom, H.Y.: Asynchronous Recovery Scheme Based on Optimistic Message Logging for Mobile Computing Systems, *Proc. International Conference on Distributed Computing Systems*, pp.436–443 (2000).
22) Pradhan, D.K., Krishna, P. and Vaidya, N.H.: Recoverable Mobile Environment: Design and Trade-off Analysis, *Proc. 26th International Symposium on Fault Tolerant Computing*, pp.16–25 (1996).
23) Prakash, R. and Singhal, M.: Low-cost Checkpointing and Failure Recovery in Mobile Computing Systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.7, No.10, pp.1035–1048 (1996).
24) Sharma, D.D. and Pradhan, D.K.: An Efficient Coordinated Checkpointing Scheme for Multicomputers, *Proc. IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems*, pp.36–42 (1994).
25) Silva, L.M.E. and Silva, J.: Global Checkpointing for Distributed Programs, *Proc. 11th International Symposium on Reliable Distributed Systems*, pp.155–162 (1992).
26) Standard IEEE802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (1999).
27) Standard IEEE802.15.1: IEEE Std. 802.15.1 - 2005: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs) (2005).
28) Xu, J., Netzer, R.H.B. and Mackey, M.: Sender-based Message Logging for Reducing Rollback Propagation, *Proc. 7th IEEE Symposium on Parallel and Distributed Processing*, pp.602–609 (1995).
29) Yao, B., Ssu, K.F. and Fuchs, W.K.: Message Logging in Mobile Computing, *Digest of Papers of the 29th Annual International Symposium on Fault-Toleran Computing*, pp.294–301 (1999).
30) Zeng, X., Bagrodia, R. and Gerla, M.: GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks, *Proc. IEEE Workshop on Parallel and Distributed Simulation*, pp.182–185 (1998).

(平成 19 年 5 月 19 日受付)
(平成 19 年 11 月 6 日採録)

小野 真和（学生会員）

昭和 55 年生．平成 17 年東京電機大学大学院理工学研究科情報システム工学専攻修了．同年東京電機大学大学院先端科学技術研究科先端技術創成専攻（博士後期課程）入学．現在 3 年次在学．平成 16 年第 66 回情報処理学会全国大会大会奨励賞，平成 17 年第 67 回情報処理学会全国大会学生奨励賞受賞．IEEE，電子情報通信学会各会学生会員．

桧垣 博章（正会員）

昭和 42 年生．平成 2 年東京大学工学部計数工学科卒業．同年日本電信電話（株）NTT ソフトウェア研究所入所．平成 8 年東京電機大学理工学部経営工学科助手．平成 11 年東京電機大学理工学部情報システム工学科講師．平成 12 年同助教授．平成 19 年より東京電機大学未来科学部ロボット・メカトロニクス学科准教授．博士（工学）．分散システム，分散アルゴリズム，フォールトトレラントネットワーク，モバイルアドホックネットワークの研究に従事．平成 7 年情報処理学会全国大会奨励賞，平成 9 年情報処理学会山下記念研究賞，平成 17 年情報処理学会学会活動貢献賞受賞．IEEE，ACM，電子情報通信学会，電気学会各会員．