社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－DPS－125 (17)
2005－EIP－ 29 (17)
2005／11／15

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

東京電機大学 理工学部 情報システム工学科
小野 真和　桧垣 博章　足立 暁生
E-mail: {masa,hig}@higlab.k.dendai.ac.jp, adachi@k.dendai.ac.jp

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プロトコルが提案されてきた. チェックポイントプロトコルでは, 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の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, 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. また, メッセージの配送は送信元のアプリケーションから送信先のアプリケーションまで直接配送可能であるとしている. 現在まで, 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, メッセージがマルチホップで配送されるようなアドホックネットワークに注目し, 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プロトコルを提案してきた. また, 従来の固定ネットワークにおける手法ではネットワーク全体で同期を行ない, チェックポイントを設定するためのコーディネータを決定しているが, トポロジが変化しやすいアドホックネットワークではコーディネータの交代が多く発生し, 新たなコーディネータを決定するための通信オーバヘッドが大きくなるという問題がある. そのため, 本提案ではチェックポイントプロトコルの開始を任意のコンピュータとし, コーディネータを利用したネットワーク全体での同期を必要としない手法としている. 前提として, 複数のコンピュータが同時にチェックポイントの実行を開始しないとしていたが, 本論文では, その前提がなく, 複数のコンピュータによって同時に異なるチェックポイントプロトコルを開始するような環境におけ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について提案する.

Checkpoint Protocol for Mobile Ad Hoc Network

Masakazu Ono,　Hiroaki Higaki and Akeo Adachi
Department of Computers and Systems Engineering
Tokyo Denki University
E-mail: {masa,hig}@higlab.k.dendai.ac.jp, adachi@k.dendai.ac.jp

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 sender and a receiver 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 storage 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 intermediate mobile computer on its transmission route. Here, communication overhead for taking 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 [3] や HIPERLAN [1],Bluetooth [2] 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の研究も進んでいる. 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 [5,14] 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 [9] は安定記憶 [13] がネットワーク上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期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通信リンクが狭帯域幅で低信頼であるため, このネットワークを介した同期に要する通信オーバヘッドが大きい問題を解決する必要がある. また, メッセージの配送が送信元のアプリケーションから送信先のアプリケーションまで複数の移動コンピュータを経由して配送される, マルチホップ配送を行なっている. このとき, 移動コンピュータの移動などによってメッセージの配送経路が変

わり,従来とは異なった条件で一貫性のないメッセージが発生する.そこで,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の実現に起因する通信オーバヘッドを回避する新たなチェックポイントプロトコルを提案する.

2 従来手法

2.1 チェックポイントプロトコル

アドホックネットワーク $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ とは,移動コンピュータの集合 $\mathcal{V}$ と互いに直接メッセージを交換することが可能な移動コンピュータ M_i, $M_j \in \mathcal{V}$ の間の双方向リンク $\langle M_i, M_j \rangle$ の集合 $\mathcal{E}$ で定まるネットワークである.一般に,ネットワーク環境において,チェックポイントプロトコルによって各移動コンピュータ $M_i \in \mathcal{V}$ が設定した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 $C_{\mathcal{V}}$ が一貫性を持つとは,次の性質を満たすことをいう [6].

[定義]
1) 送信元移動コンピュータ M_s から送信先移動コンピュータ M_r へ配送されるメッセージ m が紛失メッセージであるとは,グローバルチェックポイント $C_{\mathcal{V}}$ に対して, $Send(m)$ が c_s に先行し, c_r が $Receive(m)$ に先行することである.なお, $Send()$ と $Receive()$ は,アプリケーション層におけるメッセージ送受信イベントである.
2) メッセージ m が孤児メッセージであるとは, $C_{\mathcal{V}}$ に対して, c_s が $Send(m)$ に先行し, $Receive(m)$ が c_r に先行することである.
3) 紛失メッセージ,孤児メッセージを含まないグローバルチェックポイントは,一貫性が保たれているという.□

ただし,紛失メッセージをリカバリ回復時に再送信することができれば,システム状態の一貫性を維持することが可能である.そこで,一貫性のあるグローバルチェックポイントを以下のように再定義する.

[定義]
4) 一貫性のあるグローバルチェックポイントとは,孤児メッセージを含まず,すべての紛失メッセージをリカバリ回復時に再送信可能であるものである.□

この定義にしたがって,M_r で紛失メッセージ m を記憶,保存するプロトコルとして [8] がある.

従来のチェックポイントプロトコルにおいては,m が $C_{\mathcal{V}}$ に対する紛失メッセージや孤児メッセージとなることを M_r でのみ判定することを前提としている.そのため,これらの発生を回避するには,システム全体での同期を必要としていた.例えば,Koo [12] のプロトコルにおいては,ネットワーク内のコーディネータとなるコンピュータが同期メッセージを送信し,チェックポイント要求メッセージ $CReq$ を受信してから,チェックポイント終了メッセージ $CFin$ を受信するまでの間,アプリケーションメッセージの送信を禁止している.ところが,アドホックネットワークにおいては,無線通信の狭帯域幅,無線信号の減衰と複数無線信号の衝突による低信頼性,無線通信帯域の予約における競合,マルチホップ配送による伝達遅延などのために移動コンピュータ間の同期に要する通信オーバヘッドが大きくなる.この結果,アプリケーションの実行を一時停止する時間が長くなる,すなわち,チェックポイントプロトコルの開始から終了までの時間が長くなるという問題が発生する.また,移動コンピュータの移動などにより,コーディネータとなる移動コンピュータがネットワーク内に存在しなくなり,新たなコーディネータを決定しなければならなくなる.このとき,コーディネータを選択するための選択アルゴリズムが同期のために通信を行なわなければならず,そのための通信オーバヘッドも大きくなるといった問題も発生する.本論文で提案するプロトコルでは,アドホックネットワークにおいて,m が紛失メッセージあるいは孤児メッセージとなる可能性を m の配送経路上にある移動コンピュータが判定し,必要に応じて m を記憶したり,m の転送を遅延させたりすることによってこの問題を解決する.ここでは,隣接する移動コンピュータ間における同期のみが必要であることから,アプリケーションの停止時間を短縮することが可能である.また,ネットワーク全体での同期を必要としないため,コーディネータを必要とせず,コーディネータを選択するためのオーバヘッドも削減することができる.

2.2 モバイルチェックポイントプロトコル

モバイルチェックポイントプロトコルの実現にあたり,論文 [15] では,移動コンピュータを含むネットワークを以下の 4 つのモデルに分類している.

1) Centralized Networks
2) Cell–Dependent Infrastructured Networks
3) Cell–Independent Infrastructured Networks
4) Ad–hoc Networks

1)–3) は,ネットワークの構成要素に固定コンピュータを含んでいる.そこで,固定コンピュータに実現した安定記憶に移動コンピュータの状態情報を保存することにより,チェックポイントを設定することができる.論文 [10] では,同期チェックポイント手法と非同期チェックポイント手法を組み合わせた複合チェックポイント手法を提案している.[10], [16] および [15] において,それぞれ 1),2) に対するプロトコルを設計している.また, [4] と [17] も 1) を対象として固定コンピュータの安定記憶を使用するプロトコルを提案している.ところが 4) においては,ネットワークの構成要素が移動コンピュータのみであることから,安定記憶の実現が困難である.そこで,各移動コンピュータのチェックポイントの設定を,複数の移動コンピュータに状態情報を記憶させることによって実現する.

3 提案プロトコル

3.1 チェックポイントプロトコル

以下の条件のもとでプロトコルを構成する.

[前提条件]
1) 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.
2) 隣接する移動コンピュータ間の通信リンクは,動的に切断および確立されることがある.

3) 各移動コンピュータは, 隣接する移動コンピュータのリストを保持している.
4) 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, 半二重通信が行なわれる. また, ユニキャスト通信は, 受信確認と再送機構により, メッセージの紛失なく実現されているものとする. □

チェックポイントプロトコルの基本形を示す. チェックポイントプロトコルの開始は, 任意の移動コンピュータが任意のタイミングで行なうことができる. チェックポイント設定要求の伝達と, チェックポイント間の同期は, 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 [7] によって実現される (図 1).

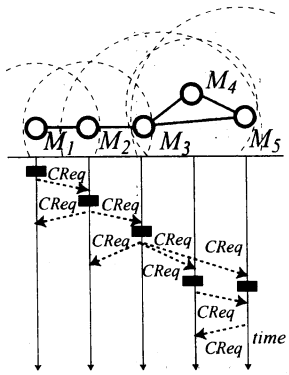

図 1: チェックポイントプロトコル

CReq を受信した移動コンピュータ M_i は, 現在の状態情報 S_i 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する. これを繰り返すことによって, 前提条件 1) 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.

ここで, 移動コンピュータ M_i の状態情報 S_i を保存する安定記憶を実現するために, S_i を複数の隣接移動コンピュータに記憶させる手法を用いる. 各移動コンピュータは, ローカルチェックポイント c_i における状態情報 S_i を獲得してから *CReq* のブロードキャストを行なうことから, S_i を *CReq* によって配送することにより, 追加のメッセージを要することなく S_i の配送が実現される.

[アドホックチェックポイントプロトコル (定義)]
M_i が開始したチェックポイントを一意に識別できるチェックポイント識別子を $\langle M_i, ID_i \rangle$ とするとき, 各移動コンピュータがこの識別子を持つ *CReq* によって設定するローカルチェックポイントを $c_j(\langle M_i, ID_i \rangle)$ とし, このときに保存する状態情報を $S_j(\langle M_i, ID_i \rangle)$ とする. また, 各コンピュータのチェックポイント $c_j(\langle M_i, ID_i \rangle)$ の集合をグローバルチェックポイント $C_\nu(\langle M_i, ID_i \rangle)$ と表わすこととする.

[アドホックチェックポイントプロトコル (基本形)]
1) 任意の移動コンピュータ M_0 が, M_0 の状態情報 S_0 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, S_0 を含み, 移動コンピュータ M_0 の *ID* と M_0 が生成した *ID* の組であるチェックポイント識別子 $\langle M_0, ID_0 \rangle$ を付加した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する. このとき, タイマ $T_0(\langle M_0, ID_0 \rangle)$ をセットする.
2) 移動コンピュータ M_i が送信したチェックポイント設定要求メッセージ $CReq(\langle M_0, ID_0 \rangle)$ を受信した移動コンピュータ M_j は, 以下の処理を行なう.
   2-1) M_i から同一のチェックポイント識別子 $\langle M_0, ID_0 \rangle$ 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 $S_i(\langle M_0, ID_0 \rangle)$ 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のチェックポイント識別子 $\langle M_0, ID_0 \rangle$ を持つ *CReq* を受信していないならば, M_j の状態情報 $S_j(\langle M_0, ID_0 \rangle)$ を獲得するとともに, $S_j(\langle M_0, ID_0 \rangle)$ を含み, 受信した *CReq* と同一のチェックポイント識別子 $\langle M_0, ID_0 \rangle$ を付与した *CReq* を M_j の無線信号到達範囲内にブロードキャストする. このとき, タイマ $T_j(\langle M_0, ID_0 \rangle)$ 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 L_j に含まれるすべての移動コンピュータから $CReq(\langle M_0, ID_0 \rangle)$ を受信する以前にタイマ $T_j(\langle M_0, ID_0 \rangle)$ が時間切れとなったならば, M_j は, 同じ $CReq(\langle M_0, ID_0 \rangle)$ を再度ブロードキャストする. □

このとき, チェックポイントプロトコルの実行と平行に送受信されたメッセージは $C_\nu(\langle M_i, ID_i \rangle)$ に対して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に保存し, リカバリ回復時に, 保存されたメッセージを再送信することによって, システム状態の一貫性を維持することができる. 一方, 孤児メッセージは, リカバリ再実行時に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障がないことから, その発生を回避しなければならない. 移動コンピュータの移動による通信路の切断と接続が発生しない場合には, 1 つのグローバルチェックポイント $C_\nu(\langle M_i, ID_i \rangle)$ に対して以下の性質が成り立つ.

[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,
   $receive(m_l) \rightarrow c_i(\langle M_k, ID_k \rangle) \rightarrow send(m_l)$ が成り立つ.
   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る. また, $\rightarrow$ は, イベント間の happened–before の関係を表すものとする.
   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, $send(m_l) \rightarrow c_i(\langle M_k, ID_k \rangle)$ かつ $c_j(\langle M_k, ID_k \rangle) \rightarrow receive(m_l)$ が成り立つ.
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, $c_i(\langle M_k, ID_k\rangle) \rightarrow send(m_o)$ かつ $receive(m_o) \rightarrow c_j(\langle M_k, ID_k\rangle)$ が成り立つ. □

性質 1) により, 紛失メッセージとなる可能性があるメッセージ 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, この検出が送信先移動コンピュータ M_r でのみ検出可能であるならば (例えば, 論文 [6], [12] では,M_r で m_l が紛失メッセージとなることを検出している.), 図 2 に示すように,M_r が m_l を受信した時点, すなわち $Receive(m_l)$ においては,M_r がすでに $CReq$ 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 を隣接移動コンピュータ

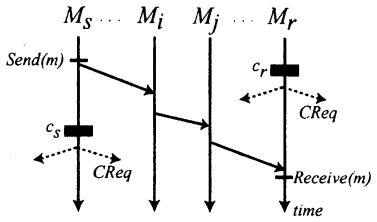

図 2: 紛失メッセージの検出とチェックポイント

に記憶させるために,m_l を含む制御メッセージをブロードキャストする必要がある. しかし, これによって, 以下の問題が発生する.

(1) M_r がリカバリで必要とする情報を隣接移動コンピュータに記憶させるために M_r が送受信するメッセージが増加する.
(2) M_r の状態情報 S_r を保存した隣接移動コンピュータがこの制御メッセージの送信時点においても M_r の無線信号到達範囲内に存在することは保証できない. すなわち,S_r のみを持つ移動コンピュータ,m_l のみを持つ移動コンピュータが発生することがある. この結果, リカバリで必要とする情報が複数の移動コンピュータに分散することによって, リカバリ時に送受信されるメッセージ数が増加する.
(3) リカバリ回復時の再送信を必要とするすべての紛失メッセージを隣接移動コンピュータに記憶し, チェックポイントプロトコルが終結したことを各移動コンピュータが検出するためには, 新しい同期メッセージの導入が必要となる.

そこで, 紛失メッセージとなる可能性のあるメッセージ m_l を $CReq$ を送信する前に検出可能な移動コンピュータの存在が不可欠である.

論文 [21] では, 同時にたかだか 1 つのチェックポイントが実行されるとき, 性質 1-a) および 1-b) をみたすメッセージを中継する移動コンピュータにおいて $CReq$ を送信する前に検出し, 安定記憶に保存できることを示している (図 3(a),(b)).

ここで, 複数のチェックポイントプロトコルが同時に動作する場合を考える. 論文 [21] では, 紛失となる可能性のあるメッセージが複数回検出され, かつ保存された場合, リカバリ時にそのメッセージが再送されると送信先の移動コンピュータで多重受信が発生し, グローバル

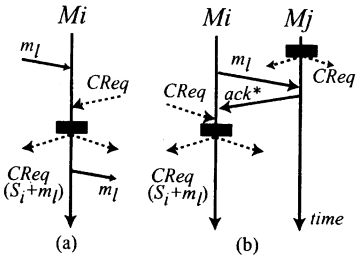

図 3: 紛失となる可能性のあるメッセージの検出と保存

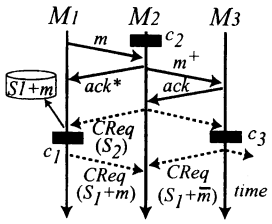

図 4: 紛失となる可能性のあるメッセージの多重保存回避

チェックポイント C_ν の一貫性がない状態になるという問題が発生するため, 複数回紛失となる可能性が検出されたとしても一度だけしか保存されない手法を提案している (図 4). しかし, 複数のチェックポイントプロトコルが同時に動作する場合を考えたとき, 一度だけしか保存しなかったためにリカバリ後に再送されない場合が考えられる.

たとえば, 図 5 のような場合を考えたとき, アプリケーションメッセージ m は M_1 と M_2 間において $C_\nu(\langle M_i, ID_i\rangle)$ に対して紛失となる可能性を検出され保存される. また,M_2 と M_3 においては $C_\nu(\langle M_j, ID_j\rangle)$ に対して紛失となる可能性を検出されるが, 従来手法 [21] では複数のチェックポイントにおいて紛失となる可能性を検出することができないため, ここで m は保存されない (図 5(a)). もし,M_3 が m を保存せず, $C_\nu(\langle M_j, ID_j\rangle)$ から処理を再開した場合, m が再送されないため, 正しく処理を再開できない問題がある.

そこで, 同時に複数のチェックポイントプロトコルが実行されたとしても, 孤児メッセージおよび, 紛失メッセージとなる可能性のあるメッセージを検出できるようにする必要がある. まず, 移動コンピュータはチェックポイント設定時に以下の処理を行なう.

[チェックポイント設定時の移動コンピュータの処理]
移動コンピュータ M_i がチェックポイント識別子 $\langle M_j, ID_j\rangle$ が付加された $CReq$ を受信し, ローカルチェックポイント $c_i(\langle M_j, ID_j\rangle)$ を設定したとき, チェックポイント識別子を, チェックポイント実行履歴 C_i に追加する. もし, C_i に $\langle M_j, ID_j\rangle = \{\langle M_k, ID_k\rangle | M_j = M_k, ID_k < ID_j\}$ となるような $\langle M_k, ID_k\rangle$ が存在したとき, それは削除する.

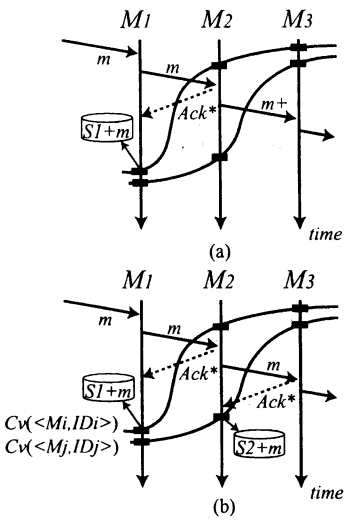

図 5: 複数のグローバルチェックポイントに対して紛失となる可能性が検出される場合

また, 各移動コンピュータは送信および転送を行なうメッセージに対し, 以下の処理を行なうこととする.

[メッセージ送信時の処理 (アプリケーション)]
アプリケーションメッセージを送信する移動コンピュータ M_i はメッセージを送信するときに M_i が保存しているチェックポイント実行履歴 $\mathcal{C}_i$ を $\mathcal{C}_{app_i}$ として m に付加する.

[メッセージ転送時の処理 (ネットワーク)]
メッセージを転送する移動コンピュータ M_i はメッセージを転送するときに M_i が保存しているチェックポイント実行履歴 $\mathcal{C}_i$ を $\mathcal{C}_{net_i}$ として m に付加する.

転送先コンピュータ M_j はメッセージに付加された転送元 M_i の $\mathcal{C}_{net_i}$ を参照し, 自身が保持する $\mathcal{C}_j$ と比較することにより, チェックポイント識別子 $\langle M_k, ID_k\rangle \in \mathcal{C}_j$ によって定められるグローバルチェックポイント $C_\mathcal{V}(\langle M_k, ID_k\rangle)$ に対して紛失となる可能性があるかどうかを検出することができる.

たとえば, 図 5(b) において,M_2 は M_1 から転送されてきた m に付加された $\mathcal{C}_{net_1}$ と M_2 のチェックポイント実行履歴 $\mathcal{C}_2$ を比較する. もし, チェックポイント識別子 $\langle M_i, ID_i\rangle$ が $\mathcal{C}_{net_1}$ に含まれないかその古いチェックポイントしかなく, かつ, $\langle M_i, ID_i\rangle$ が $\mathcal{C}_2$ に含まれるとき, M_2 は転送されてきたメッセージ m がグローバルチェックポイント $C_\mathcal{V}(\langle M_i, ID_i\rangle)$ に対して紛失メッセージとなる可能性があることを検出でき, 保存できる. 同様に,M_3 においても m が $C_\mathcal{V}(\langle M_j, ID_j\rangle)$ に対して紛失メッセージとなる可能性があることを検出でき, 保存することができる (図 5(b)).

以上のことから, 複数のチェックポイントが同時に実行される場合においてもどのグローバルチェックポイントに対しても紛失メッセージとなる可能性があるメッセージを検出し, $CReq$ を送信する前のコンピュータに保存することができる.

次に, 孤児メッセージの検出と発生回避について考える.

孤児メッセージは, アプリケーションメッセージの送信先コンピュータが受信したメッセージが孤児メッセージであると判断したとき, アプリケーションにおける受理を延期し, メッセージが $C_\mathcal{V}(\langle M_i, ID_i\rangle)$ に対して孤児メッセージとなることを回避している.

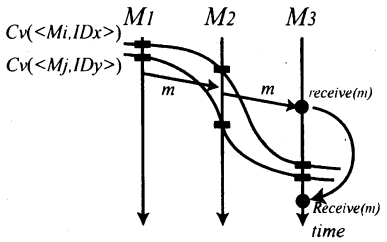

図 6: 孤児メッセージ発生回避

孤児メッセージの検出は, アプリケーションメッセージの送信先移動コンピュータで行なう. 受信した孤児メッセージかどうかの判断は, 送信元移動コンピュータがメッセージ送信時に付加したチェックポイント設定履歴 $\mathcal{C}_{app_i}$ と送信先コンピュータのチェックポイント設定履歴 $\mathcal{C}_j$ を比較することによって行なう.

図 6 において, メッセージ m の送信先が M_3 であるとする. このとき,M_3 は自身の $\mathcal{C}_3$ と m に付加されている $\mathcal{C}_{app_i}$ を比較する. ここで,M_3 はチェックポイント $c_3(\langle M_i, ID_i\rangle)$ および $c_3(\langle M_j, ID_j\rangle)$ の設定を行なっていないことが分かり, m が $C_\mathcal{V}(\langle M_i, ID_i\rangle)$ と $C_\mathcal{V}(\langle M_j, ID_j\rangle)$ に対して孤児メッセージであることが分かる. したがって,m は $c_3(\langle M_i, ID_i\rangle)$, $c_3(\langle M_j, ID_j\rangle)$ の設定が終わるまでアプリケーションにおける受理が延期される.

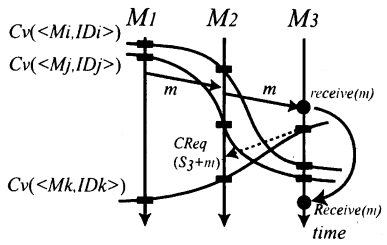

図 7: 孤児メッセージ発生回避によって発生する紛失メッセージ

ここで, メッセージの受理を延期しているとき, 図 7 のような状況が発生することが考えられる. アプリケーションメッセージ m の送信先が M_3 であり, M_3 は $C_\mathcal{V}(\langle M_j, ID_j\rangle)$ および $C_\mathcal{V}(\langle M_k, ID_k\rangle)$ に対して m が孤児メッセージであることを検出し, アプリケーションでの受理をチェックポイント $c_3(\langle M_j, ID_j\rangle)$, $c_3(\langle M_k, ID_k\rangle)$

が設定される後まで延期している．もし，このとき，M_3 において $c_3(\langle M_i, ID_i\rangle)$ が実行され，$C_V(\langle M_i, ID_i\rangle)$ が決定できたとき，メッセージ m は $C_V(\langle M_i, ID_i\rangle)$ に対して紛失メッセージとなってしまう．

紛失メッセージは安定記憶に保存され，$C_V(\langle M_i, ID_i\rangle)$ から処理が再開されるときに m の再送信が行なわなければならない．ここで，M_3 は $c_3(\langle M_i, ID_i\rangle)$ を設定する際に m を $S_3(\langle M_i, ID_i\rangle)$ に付加することが可能であり，新たな同期メッセージを必要とせず m を安定記憶には位置することができ，紛失メッセージの発生を回避することができる．

また，$c_3(\langle M_j, ID_j\rangle)$，$c_3(\langle M_k, ID_k\rangle)$ が設定されるまで m の受理を遅らせれば，孤児メッセージとなることも回避できる．

4 まとめと今後の課題

本論文では，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を示した．論文 [21] において提案されている手法において，複数のチェックポイントが同時に発生する場合について検討し，そのような場合におけるチェックポイント設定手法について提案した．このとき，各移動コンピュータが設定したチェックポイントの履歴を保存し，比較することにより同期メッセージを必要とせず，ホップバイホップで紛失となる可能性のあるメッセージの検出や保存を行なうことができる．また，あるグローバルチェックポイントに対して孤児メッセージのとなったメッセージのアプリケーションにおける受理を延期している時に，そのメッセージが他のグローバルチェックポイントに対して紛失メッセージとなってしまう場合の解決法について示した．

今後は複数同時に行なわれたチェックポイントプロトコルで設定したチェックポイントを利用して一貫性のある状態から処理を再開する手法について検討する．

なお，本研究は東京電機大学総合研究所 研究課題 Q05J-05 として行なわれた．

参考文献

[1] "Radio Equipment and Systems (RES); HIPERLAN," ETSI,Functional Specifications (1995).

[2] "Wireless (MAC) and (PHY) Specifications for Wireless Personal Area Networks," Standard IEEE 802.15.1 (2002).

[3] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," Standard IEEE 802.11 (1999).

[4] Acharya, A. and Badrinath, B.R., "Checkpointing Distributed Applications on Mobile Computers," Proc. of 3rd International Conference on Parallel and DistributedInformation Systems, pp. 73–80 (1994).

[5] Callaway, E.M., "Wireless Sensor Networks," Auerbach Publications (2003).

[6] Chandy, K.M. and Lamport, L., "Distributed Snapshots: Determining Global States of Distributed Systems," ACM Trans. on Computer Systems, Vol. 3, No. 1, pp. 63–75 (1985).

[7] Corson, M.S. and Ephremides, A., "A Distributed Routing Algorithm for Mobile Wireless Networks," ACM Journal of Wireless Networks, vol. 1, No. 1, pp. 61–81 (1995).

[8] Elnozahy, E.N. and Zwaenepoel, W., "On the use and Implementation of Message Logging," Proc. of the Fault-Tolerant Computing Symposium, pp. 298–307(1994).

[9] Gendelman, E., Bic, L.F. and Dillencourt, M.B., "An efficient checkpointing algorithm for distributed systems implementing reliable communication channels," Proc. of 18th International Symposium on Reliable Distributed Systems, pp. 290–291 (1999).

[10] Higaki, H. and Takizawa, M., "Checkpoint-Recovery Protocol for Reliable Mobile Systems," Proc. of the 17th International Symposium on Reliable Distributed Systems, pp.93–99 (1998).

[11] Johnson, D.B., Maltz, D.A., Hu, Y.C., and Jetcheva, J.G., "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," Internet Draft, draft-ietf-manet-dsr-04.txt (2000).

[12] Koo, R. and Toueg, S., "Checkpointing and Rollback-Recovery for Distributed Systems," IEEE Trans. on Software Engineering, Vol. SE-13, No. 1, pp. 23–31 (1987).

[13] Lampson, B.W., Paul, M. and Siegert, H.J., "Distributed Systems – Architecture and Implementation," Springer-Verlag, pp. 246–265 (1981).

[14] Lesser, V., Ortiz, C.L. and Tambe, M., "Distributed Sensor Networks," Kluwer Academic Publications (2003).

[15] Miyazaki, M., Morita, Y. and Higaki, H., "Hybrid Checkpoint Protocol for Mobile Networks with Unreliable Wireless Communication Channels," Proc. of the 2nd Asian International Mobile Computing Conference, pp.164–171 (2002).

[16] Morita, Y. and Higaki, H., "Checkpoint-Recovery for Mobile Computing Systems," Proc. of the 21st Internation Conference Distributed Computing Systems Workshops, pp.479–484 (2001).

[17] Neves, N. and Fuchs, W.K., "Adaptive Recovery for Mobile Environments," Communications of the ACM, Vol. 40, No. 1, pp. 69–74 (1997).

[18] Perkins, C.E. and Royer, E.M., "Ad-hoc On-Demand Distance Vector Routing," Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 99–100 (1999).

[19] Strom, R. and Yemini, S., "Optimistic Recovery in distributed systems," ACM Trans. on Computer Systems, Vol. 3, No. 3, pp. 204–226 (1985).

[20] Yao, B. and Fuchs, W.K., "Message logging optimization for wireless networks," Proc. of the 20th International Symposium on Reliable Distributed Systems, pp. 182–185 (2001).

[21] 小野, 桧垣, "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," 情報処理学会マルチメディア通信と分散処理研究会, 情処研報, Vol.2005, No.58, pp.13–18 (2005).