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

2004－MBL－28 (23)
2004／3／5

無線マルチホップ型ネットワークにおける状態情報の分散配置による耐故障性の実現手法

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

ネットワーク環境においてミッションクリティカルアプリケーションを実現する手法として、チェックポイントリカバリプロトコルがある。従来の有線ネットワークを対象としたプロトコルでは、状態情報を格納するための安定記憶が存在することと、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って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) が検出、回避できる程度に十分な通信帯域幅が存在することが前提となっている。本論文では、これらの前提が成立しない無線マルチホップ型ネットワークにおけるチェックポイントプロトコルを提案する。k–同時故障への耐性を実現するために、各移動コンピュータの状態情報は、無線マルチホップ通信で到達可能な無線基地局の安定記憶か k 台の近隣移動コンピュータの揮発性記憶かのいずれかに格納する。このとき、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状態情報の一部として記憶することにより、状態情報とメッセージログを同一の移動コンピュータに同時に保存することができる。これによって、チェックポイントプロトコルの開始から終了までに要する時間を短縮することができる。

Checkpoint Protocol for k–Resilient Wireless Multi-hop Networks

Masakazu Ono　Tatsuya Hirakawa　Hiroaki Higaki
Department of Computers and Systems Engineering
Tokyo Denki University
E-mail: {masa,tatuya,hig}@higlab.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 wireless multi-hop networks without 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 a stable storage in a base station or volatile storages in multiple mobile computers for k-resilience. A candidate of a lost message is detected and stored by an 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する無線LAN技術の研究開発と普及が進んでいる。現在の無線LANの適用形態は、有線ネットワークと無線ネットワークとの間のゲートウェイとして無線基地局が介在し、無線基地局とその無線信号到達範囲内に存在する移動コンピュータとが直接メッセージを交換するインフラストラクチャネットワークが中心となっている。ここでは、メッセージのルーティングは、有線ネットワークに接続されたルータの機能によって実現されている。しかし、インフラストラクチャネットワークでは、他のコンピュータと通信可能な移動コンピュータは、いずれかの無線基地局の無線信号到達範囲内に存在するものに限られている。いずれの無線基地局の無線信号到達範囲にも含まれていない移動コンピュータは、他のコンピュータと通信することができない。すなわち、ネットワークの接続性 (コネクティビティ) が低いといえる。無線マルチホップ型ネットワークにおいては、いずれの無線基地局の無線信号到達範囲にも含まれない移動コンピュータも、他のいずれかの移動コンピュータとメッセージを直接交換することが可能であるならば、このメッセージをマルチホップ配送を用いて、いずれかの無線基地局まで配送することによって、他のコンピュータとの通信が可能となる。つまり、ネットワークの接続性を高めることが可能である。ITS(高度交通システム) や自律移動型ロボット群の制御、センサネットワーク等の応用システムにおける無線マルチホップ型ネットワークの導入が考えられる。無線マルチホップ型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] では、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との同期によって検出することが可能となる

十分な帯域幅が提供されていることを前提としている。しかし、無線マルチホップ型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は狭帯域幅で低信頼であるため、同期に要する通信オーバヘッドが大きい。また、移動コンピュータの位置によって無線基地局との間の通信コスト (ホップ数) が異なる。そのため、インフラストラクチャネットワークのためのチェックポイントプロトコル [10,14,15] に基づいて、各移動コンピュータのローカルチェックポイントにおける状態情報の保存位置を無線マルチホップ配送で到達可能な無線基地局に一意に決定することは適切ではない。本論文では、グローバルチェックポイントの一貫性を失わせる可能性のあるメッセージを中継コンピュータが検出することによって、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し、移動コンピュータの状態情報を近隣の複数の移動コンピュータに分散配置することによって、k-同時故障に耐えることができる状態情報の保存に要する通信オーバヘッドを削減する新しい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

無線マルチホップ型ネットワーク $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ における移動コンピュータの集合 $\mathcal{V}$ は、固定コンピュータの集合 $\mathcal{F}$ 、無線基地局の集合 $\mathcal{B}$ 、移動コンピュータの集合 $\mathcal{M}$ からなり、$\mathcal{V} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{M}$ である。また、$\mathcal{E}$ は、互いに直接メッセージを交換することが可能なコンピュータ N_i, $N_j \in \mathcal{V}$ の間の双方向リンク $\langle N_i, N_j \rangle$ の集合である。ここで、すべての固定コンピュータ、無線基地局は、有線ネットワークのルーティング機能によって互いに直接メッセージを交換することができる。すなわち、$\forall N_i, \forall N_j \in \mathcal{F} \cup \mathcal{B}$ について $\langle N_i, N_j \rangle \in \mathcal{E}$ である。一方、移動コンピュータ間および移動コンピュータと基地局との間のメッセージ交換は、互いの無線信号到達範囲内に存在するときにのみ可能である。以降では、$N_i, N_j \in \mathcal{V}$ について $\langle N_i, N_j \rangle \in \mathcal{E}$ であるとき、N_j は N_i の隣接コンピュータであるという。

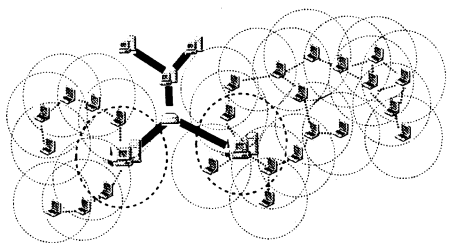

図 1: 無線マルチホップ型ネットワーク

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コンピュータ $N_i \in \mathcal{V}$ 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) 送信元コンピュータ N_s から送信先コンピュータ N_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、N_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 N_r でのみ判定する。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 [11] のプロトコルにおいては、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することによって、孤児メッセージの発生を回避し、すべての紛失メッセージを送信先コンピュータで検出、保存することを可能としている。ところが、無線マルチホップ型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および移動コンピュータと無線基地局との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。本論文で提案するプロトコルでは、無線マルチホップ型ネットワークにおいて、m が紛失メッセージとなる可能性を m の配送経路上にある中継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。

2.2 モバイルチェックポイントプロトコル

移動コンピュータを含むネットワーク環境を対象とした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 [14] 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) 集中型インフラストラクチャネットワーク
(2) 自律型インフラストラクチャネットワーク
(3) 無線マルチホップ型ネットワーク
(4) アドホックネットワーク

(1) と (2) では、すべての移動コンピュータがいずれかの無線基地局と直接メッセージを交換することが可能である。したがって、無線基地局に安定記憶を実現し、各移動コンピュータのローカルチェックポイントにおける状態情報を保存すればよい。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]. [14] 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 [16] も (1) を対象として無線基地局の安定記憶を使用する同期チェックポイントプロトコルを提案している。ところが (3) においては、無線基地局へのメッセージ配送にマルチホップを要する移動コンピュータも存在する。この場合、無線基地局に状態情報を記憶するよりも、近隣の移動コンピュータに記憶する方が要する通信オーバヘッドが小さい。k–同時故障に耐えるため、k 台の近隣移動コンピュータまたは無線基地局にローカルチェックポイントにおける状態情報を保存することによって、通信オーバヘッドと記憶容量の削減が可能となる。

3 提案プロトコル

以下の条件のもとでプロトコルを構成する。

[前提条件]

(1) 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、マルチホップ配送を用いて、いずれかの無線基地局とメッセージを交換することが可能である。
(2) 各コンピュータは、隣接コンピュータの ID を保持している。
(3) 隣接コンピュータ間の通信リンクは双方向である。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。 □

3.1 チェックポイント要求メッセージの配送

本論文で提案するプロトコルは論文 [11] と同様に $CReq$, $CRep$, $CFin$ の 3 種類の制御メッセージの交換によって実現される。チェックポイントプロトコルの開始は、有線ネットワークに接続されたいずれかのコンピュータ (以降ではコーディネータとよぶ) が任意のタイミングで行なうこととする。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 2)。

まず、すべての固定コンピュータと無線基地局がコーディネータからの $CReq$ を有線ネットワークを介して受信し、状態情報を獲得することでローカルチェックポイントを設定する。この状態情報は、自身に実現した安定記憶に格納される。一方、無線基地局は、$CReq$ を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。$CReq$ 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、S_i を含む $CReq$ を隣接コンピュータ群、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、すべてのコンピュータが $CReq$ を受信し、ローカルチェックポイントを設定することができる。

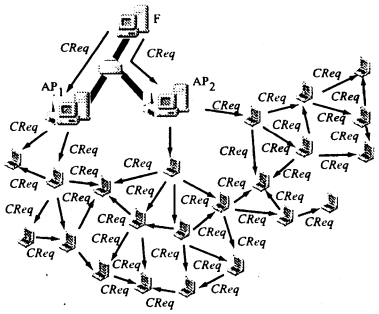

図 2: チェックポイント要求メッセージの配送

無線基地局の無線信号到達範囲内に存在する移動コンピュータ M_i から送信された $CReq$ は、S_i を含んでいる。この $CReq$ は、無線基地局によって必ず受信されるため、無線基地局の安定記憶に格納することができる。一方、無線基地局と隣接しない移動コンピュータ M_i の状態情報 S_i は、M_i の隣接移動コンピュータに記憶させることとする。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 を獲得した後に $CReq$ 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。

3.2 紛失可能メッセージの検出

$CReq$ メッセージの配送と並行に送受信されるメッセージは、紛失メッセージや孤児メッセージとなる可能性がある。紛失メッセージを、いずれかのコンピュータに格納し、リカバリ回復時に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。ここで、マルチホップ配送環境においては、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 台以上のコンピュータ N_i において、$receive(m_l) \rightarrow c_i \rightarrow send(m_l)$ 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 台のコンピュー

タ N_i,N_j において、$\langle N_i, N_j\rangle \in \mathcal{E}$ であり、$send(m_l) \rightarrow c_i$ かつ $c_j \rightarrow receive(m_l)$ 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 台のコンピュータ N_i,N_j において、$\langle N_i, N_j\rangle \in \mathcal{E}$ であり、$c_i \rightarrow send(m_o)$ かつ $receive(m_o) \rightarrow c_j$ 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コンピュータが検出することができる。この m_l を紛失可能メッセージとよぶ。もし、この検出が送信先コンピュータ N_r でのみ可能であるならば (例えば、論文 [6], [11] では、N_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、N_r が m_l を受信した時点、すなわち $Receive(m_l)$ においては、N_r がすでに $CReq$ を隣接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。

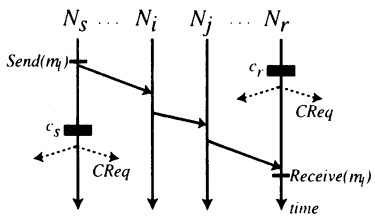

図 3: 紛失メッセージの検出とチェックポイント

この場合、m_l を隣接コンピュータ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) N_r がリカバリ回復で必要とする情報を隣接コンピュータに記憶させるために N_r が送受信するメッセージが増加する。

(2) N_r の状態情報 S_r を保存した隣接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 N_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 のみを持つコンピュータ、m_l のみを持つコンピュータが発生することがある。この結果、リカバリ回復で必要とする情報が複数のコンピュータに分散することによって、リカバリ回復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、$CReq$ を送信する前に紛失可能メッセージ m_l を検出可能なコンピュータの存在が不可欠である。ここで、条件 1-a) を満たす m_l については、N_i が m_l を $CReq$ の送信前に検出することができる。

[紛失可能メッセージの検出 (1)]

コンピュータ N_i が中継メッセージ (N_i を送信元、送信先としないメッセージ)m_l を $CReq$ の送信前に受信し、$CReq$ 送信時までに送信していないならば、m_l は紛失可能メッセージである。N_i は状態情報 S_i とともに m_l を $CReq$ に含めてブロードキャストし、これを受信した N_i の隣接コンピュータは、S_i と m_l を記憶する。□

一方、条件 (1-b) を満たすメッセージ m_l については、m_l が紛失可能メッセージであることを検出できるのは N_j であり、検出したとき N_j はすでに $CReq$ メッセージを送信済みであることがある。そこで、以下の方法により m_l の検出を N_j が行ない、N_i が $CReq$ を送信する前にその結果を N_i に通知する (図 4)。

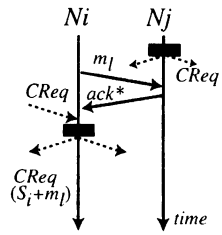

図 4: 紛失可能メッセージの検出とチェックポイント

[紛失可能メッセージの検出 (2)]

$CReq$ を送信していないコンピュータ N_i から送信されたメッセージ m_l を隣接コンピュータ N_j が $CReq$ 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 が紛失可能メッセージであることを示す情報を付加する。N_i は、これを受信したならば、状態情報 S_i とともに m_l を $CReq$ に含めてブロードキャストする。この手法を適用するために、各コンピュータが、メッセージ送信から受信確認応答メッセージの受信までの間に $CReq$ を送信することを禁止する。□

条件 (1-a)、条件 (1-b) は、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。例えば、図 5 において、N_1 では $send(m) \rightarrow c_1$ であり、N_2 では $c_2 \rightarrow receive(m)$ であることから、N_2 によって紛失可能メッセージであると判定される。この結果、m は N_1 からブロードキャストされる $CReq$ に含まれる。しかし、N_1 では $Send(m) \rightarrow c_1$ であり、N_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、グローバルチェックポイント $C_\mathcal{V} = \{c_1, c_2, c_3\}$ からリカバリしたならば、m は N_1 によって再送信されるが、N_3 では受信済みである。m が紛失可能メッセージであることを検出した N_2(m を含む $CReq$ を送信した N_1 ではない) は、m がリカバリ回復時に再送信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。m を受信した送信先コンピュータ N_3 では、m の受信が $CReq$ の送信前であるならば、再送信時に m を受信しても破棄することを示す情報を $CReq$ に含めることができる (図 5)。また、m の受信が $CReq$ の送信後であるならば、m は紛失メッセージであるので、リカバリ回復後に再送信されたものを受信する必要がある (図 6)。

また、図 7 のように複数のコンピュータが 1 つのメッセージを紛失可能メッセージであると検出すること

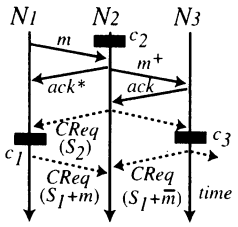

図 5: 紛失メッセージの受理回避

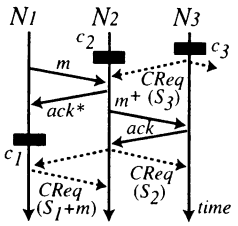

図 6: 紛失メッセージの受理

がある。例えば、m は N_2 および N_4 において紛失可能メッセージとして検出される。これは、N_1 において $send(m) \rightarrow c_1$ であり、N_2 において $c_2 \rightarrow receive(m)$ であることと、N_3 において $send(m) \rightarrow c_3$ であり、N_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、これらのコンピュータがそれぞれの *CReq* に m を含めるならば、リカバリ回復時に送信先コンピュータ N_4 は、N_1 と N_3 から再送信された 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、前段で述べた再送信時に破棄すべきメッセージであることを示す情報を用いる (図 7)。もし、この情報を含むメッセージを紛失可能メッセージであると検出しても、これを以降に送信される *CReq* に含めないことによって、最初に紛失可能メッセージを検出したコンピュータ (条件 1–a) を満たす場合) の送信する *CReq*、または、このコンピュータにこのメッセージを送信したコンピュータ (条件 1–b) を満たす場合) の送信する *CReq* にのみ m を含ませることができる。

なお、孤児メッセージとなる可能性のあるメッセージ m は、中継コンピュータで条件 2–a) に基づいて検出することが可能であるが、対処はしない。送信先コンピュータ N_r でチェックポイント c_r 後まで受信 $Receive(m)$ を遅延させることで十分である。*CReq* 配送時に状態情報とともに必要なメッセージを近隣移動コンピュータに記憶することを可能とするための、アプリケーションメッセージ配送プロトコルを以下に示す。

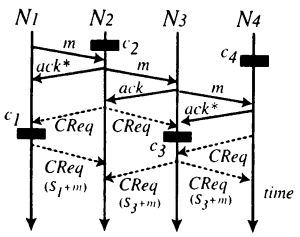

図 7: 紛失メッセージの多重検出

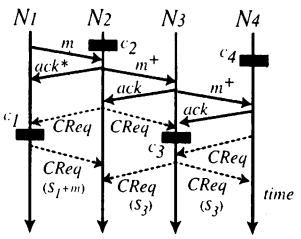

図 8: 紛失メッセージの多重検出の回避

[メッセージ通信プロトコル]

コンピュータ N_i は、アプリケーションプログラムにおける送信要求、受信要求が発生したならば、アプリケーション層の送信イベント *Send*() 、受信イベント *Receive*() を実行する。また、ネットワークからメッセージ m を受信したならば *receive*() を実行する。*send*() は、送信元コンピュータにおける *Send*() 実行時と中継コンピュータにおける *receive*() 実行時に実行される。メッセージ m には、*m.source_creq_sent*、*m.creq_sent*、*m.logged* という 3 つのフラグが含まれる。

(アプリケーション層の送信イベント $Send(m)$)

(1) $m.logged := false$ とする。
(2) N_i が *CReq* を送信済みであるならば、$m.source\_creq\_sent := true$, 未送信であるならば $m.source\_creq\_sent := false$ とする。
(3) $send(m)$ を実行する。

(アプリケーション層の受信イベント $Receive(m)$)

(1) m が $buffer_i$ に含まれないならば、$receive(m)$ の実行が終了するまで一時停止する。
(2) もし、$m.source\_creq\_sent = true$ かつ N_i が *CReq* を未送信であるならば、*CReq* の送信が終了するまで一時停止する。
(3) m を $buffer_i$ から取り出す。
(4) $m.logged = true$ かつ、N_i が *CReq* を未送信であるならば、リカバリ回復後に受信する m を破棄する情報を以降に送信される *CReq* に含める。

(ネットワーク層の送信イベント $send(m)$)

(1) N_i が *CReq* を送信済みであるならば、$m.creq\_sent := true$, 未送信であるならば $m.creq\_sent := false$ とする。
(2) *CReq* の送信を不可とする。
(3) m を送信する。
(4) $ack(m)$ を受信する。$m.logged = false$ かつ $ack(m).logged = true$ であるならば、以降に送信される *CReq* に m を含める。
(5) *CReq* の送信を可とする。

(ネットワーク層の受信イベント $receive(m)$)

(1) $m.logged = false$ かつ $m.creq\_sent = false$ かつ N_i が $CReq$ を送信済みであるならば、$m.logged := true$、$ack(m).logged := true$ として $ack(m)$ を返送する。

(2) それ以外の場合は、$ack(m).logged := m.logged$ として $ack(m)$ を返送する。

(3) m の送信先が N_i であるならば、m を $buffer_i$ に格納する。

(4) それ以外の場合は $send(m)$ を実行する。□

3.3 耐 k-同時故障の実現

無線基地局に隣接しない移動コンピュータ M_i のローカルチェックポイント c_i における状態情報 S_i と検出した紛失可能メッセージの集合 ML_i は、M_i および M_i の隣接移動コンピュータ $M_j \in Nei(M_i)$ に記憶させる。しかし、$|\ Nei(M_i)\ | + 1 > k$ であるとは限らない。ここで、M_i が最初に受信した $CReq$ を送信した隣接コンピュータを M_i^{up} とし、M_i の上流コンピュータと呼ぶ。隣接移動コンピュータ間のこの関係によって、すべての移動コンピュータは、いずれかの無線基地局を根とするツリーのノードのひとつとなる。もし、$|\ Nei(M_i)\ | + 1 \leq k$ であるならば、S_i と ML_i を M_i^{up} にユニキャスト送信する。M_i^{up} は、$Nei(M_i^{up}) - Nei(M_i)$ に S_i と ML_i を送信する。このとき、$|\ Nei(M_i)\ | + 1 + |\ Nei(M_i^{up}) - Nei(M_i)\ | \leq k$ であるならば、M_i^{up} の上流コンピュータに対して同様の手続きを要求する。これによって、S_i と ML_i を記憶した移動コンピュータ数が k を超えるか、k 以下のまま要求が無線基地局まで到達するかのいずれかである。後者の場合、S_i と ML_i を無線基地局の安定記憶に格納する。これによって k-同時故障が発生しても S_i と ML_i を M_i が獲得することが可能となる。この S_i と ML_i の上流コンピュータへの配送は、$CRep$ へのピギーバックによって実現される。

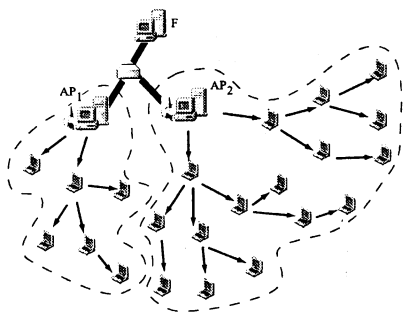

図 9: 無線基地局によって分割される木構造

[チェックポイントプロトコル]

($CReq$ の配送 (図 2))

(1) 有線ネットワークに接続された任意のコンピュータ N_0 が、N_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 とメッセージログ ML_0 を含み、N_0 が生成した ID が付与されたチェックポイント設定要求メッセージ $CReq$ を有線ネットワークに接続されたすべてのコンピュータにブロードキャストする。このとき、タイマ T_0 をセットする。

(2) $CReq$ を受信した固定コンピュータ F_i の状態情報とメッセージログを安定記憶に格納し、$CRep$ を N_0 へユニキャスト送信する。

(3) $CReq$ を受信した無線基地局 B_i は、B_i の状態情報とメッセージログを安定記憶に格納し、受信した $CReq$ と同一の ID を付与した $CReq$ を B_i の無線セル内にブロードキャストする。このときタイマ T_i をセットする。

(4) 無線基地局 B_i が、すべての隣接移動コンピュータから $CReq$ を受信する以前にタイマ T_i が時間切れとなったならば、同じ $CReq$ を再度無線セル内にブロードキャストする。

(5) 無線基地局または移動コンピュータ N_i が送信した $CReq$ を受信した移動コンピュータ M_j は、以下の処理を行なう。

(5-1) N_i から同一の ID を持つ $CReq$ を受信していないとき、受信した $CReq$ に含まれる N_i の状態情報 S_i とメッセージログ ML_i を保存する。

(5-2) M_j がいずれの隣接コンピュータからも同一の ID を持つ $CReq$ を受信していないならば、M_j の状態情報 S_j とメッセージログ ML_j を獲得するとともに、S_j と ML_j を含み、受信した $CReq$ と同一の ID 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 をセットする。

(6) 移動コンピュータ M_j がすべての隣接コンピュータから $CReq$ 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 $CReq$ を再度無線信号到達範囲にブロードキャストする。

(7) $ML_j := \phi$ とする。

($CRep$ の配送 (図 10))

(1) $CReq$ 送信時にすべての隣接コンピュータから $CReq$ を受信済みであった移動コンピュータ M_i は、$CReq$ と同一の ID を付与した $CRep$ を上流コンピュータ N_i にユニキャスト送信する。ここで、$|\ Nei(M_i)\ | + 1 \leq k$ であるならば、$\langle stored_i, S_i, ML_i \rangle$ (ただし、$stored_i := Nei(M_i)$) を $CRep$ にピギーバックする。

(2) 移動コンピュータ M_i が、上流コンピュータ N_j を除くすべての隣接コンピュータから $CRep$ を受信したならば、以下の処理を行なう。

(2-1) 受信した $CRep$ に含まれるすべての $\langle stored_l, S_l, ML_l \rangle$ の S_l と ML_l を含む $STReq$ メッセージを無線信号到達範囲にブロードキャストする。

(2-2) 各 $\langle stored_l, S_l, ML_l \rangle$ について $stored_l := stored_l \cup Nei(M_l)$ とする。$stored_l \leq k$ であるすべての $\langle stored_l, S_l, ML_l \rangle$ を含み、受信した $CRep$ と同一の ID を付与した $CRep$ を上流コンピュータへユニキャスト送信する。

(3) $STReq$ を受信した移動コンピュータは、このメッセージに含まれる S_l, ML_l のうち、まだ保存して

いないものを新たに保存する。

(3) すべての隣接移動コンピュータから $CRep$ を受信した無線基地局は、受信した $CRep$ に含まれるすべての S_i と ML_i を安定記憶に保存し、同一の ID を付与した $CRep$ を N_0 へユニキャスト送信する。

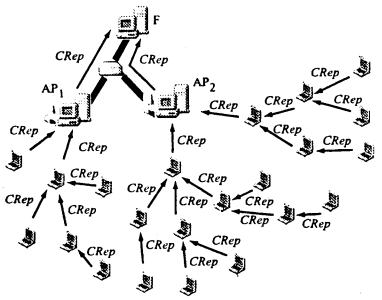

図 10: $CRep$ メッセージの配送

($CFin$ の配送 (図 11))

(1) 有線ネットワークに接続されたすべてのコンピュータから $CRep$ を受信した N_0 は、同一の ID を付与した $CFin$ を有線ネットワークに接続されたすべてのコンピュータにブロードキャストする。

(2) $CFin$ を受信した固定コンピュータ F_i は、終了する。

(3) $CFin$ を受信した無線基地局 B_i は、受信した $CFin$ と同一の ID を付与した $CFin$ を B_i の無線セル内にブロードキャストする。このとき、タイマ T_i をセットする。

(4) 無線基地局 B_i がすべての隣接移動コンピュータから $CFin$ を受信する以前にタイマ T_i が時間切れとなったならば、B_i は同じ $CFin$ を再度無線セル内にブロードキャストする。

(5) 無線基地局または移動コンピュータ N_i が送信した $CFin$ を受信した移動コンピュータ M_i は、受信した $CFin$ と同一の ID を持つ $CFin$ を受信していないならば、同一の ID を付与した $CFin$ を M_i の無線信号到達範囲内にブロードキャストする。□

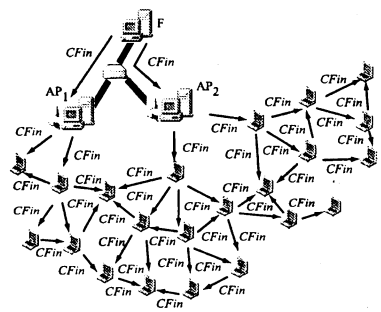

図 11: $CFin$ メッセージの配送

[耐 4–同時故障の実現例]

図 12 のように、移動コンピュータ M_4 において耐 4-同時故障を実現する場合を考える。M_4 が $CReq$ を送信することによって、M_4 の状態情報 S_4 は M_4 およびその隣接移動コンピュータである M_2,M_3,M_5 の 4 台の移動コンピュータに格納されることとなる。

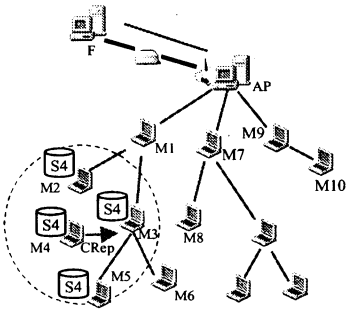

図 12: $CReq$ 配送後の S_4 の配置

しかし、M_4 を含む S_4 を保持した移動コンピュータが同時に故障した場合、リカバリを行なうことができなくなってしまう。そこで、図 12 のように M_4 は $CRep$ メッセージを上流の移動コンピュータである M_3 に送信する。$CRep$ を受信した M_3 においては、M_4 のみで耐 4–同時故障を実現できないこと判断することができる。M_3 は、自身の隣接移動コンピュータである M_1,M_6,M_8 に対して S_4 をブロードキャストする (図 13)。この結果、S_4 は 7 台の移動コンピュータが格納することなり、耐 4–同時故障が実現される。

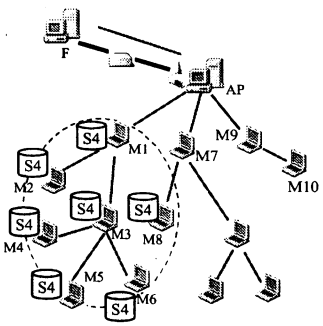

図 13: $CRep$ を利用した格納先の追加

また、M_{10} の状態情報 S_{10} は M_{10} および、その隣接移動コンピュータ M_9 の 2 台の移動コンピュータのみで保存されるため、耐 4–同時故障を実現するためには他の移動コンピュータに保存する必要がある。M_{10} は M_9 に対して $CRep$ を送信し、M_9 は自身の隣接移動コンピュータに対して S_{10} をブロードキャストする。このとき、安定記憶を持つ無線基地局 AP が M_9 の無線信号到達範囲内に存在することから、AP も S_{10} を受信する (図 14)。ブロードキャストが終了した時点で M_7,M_9,M_{10},AP の 4 台のコンピュータで S_{10} が保存されることになるが、AP の安定記憶で保存されることから、耐 4–同時故障が実現されている。□

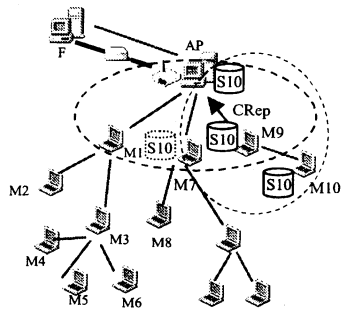

図 14: 無線基地局 AP への S_{10} の格納

4 評価

提案した耐 k-同時故障を保証したコンピュータの状態情報分散配置手法の性能をシミュレーションによって評価する。ここでは、40 台の移動コンピュータを 600m×600m の領域に配置し、基地局数を 1〜6 台の範囲とした場合の移動コンピュータの状態情報配置量を、状態情報をすべて基地局に配置する従来手法と提案手法とで比較する。移動コンピュータの位置は一様分布による乱数で決定した。無線信号の到達範囲は、移動コンピュータ、基地局ともに半径 100m の円で固定した。また、各移動コンピュータの状態情報の大きさは 32Mbyte で一定としている。図 15 は、$k = 4$ として 100 回シミュレーションを行なった平均値を示している。基地局の密度が十分に高い場合、すなわち、ほとんどの移動コンピュータが基地局と直接通信可能となる場合には、従来手法の方が高い性能を示す可能性があるものの、無線マルチホップ型ネットワークが対象とする多くの場合においては提案手法によって状態情報の配送量が削減されることが分かる。

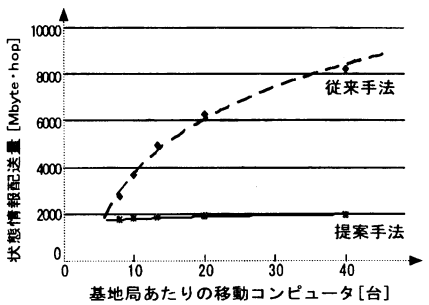

図 15: 同時故障数 $k = 4$ における状態情報配送量の比較

5 まとめと今後の課題

本論文では、無線マルチホップ型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。紛失メッセージとなる可能性のあるメッセージを、エンド-エンドではなく、ホップバイホップで検証し、メッセージログに保存する機構を導入することにより、各移動コンピュータが隣接移動コンピュータに対して、状態情報とメッセージログを含むチェックポイント要求メッセージを一度だけ送信することにより、同期オーバヘッドを削減することができる。今後は、リカバリプロトコルを設計し、提案プロトコルによるオーバヘッドの削減効果を評価する。

参考文献

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

[2] "The Official Bluetooth Wireless Info Site," http://www.bluetooth.com .

[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] 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).

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

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

[14] 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).

[15] 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).

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

[17] 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).