モーバイルコンピューティング 7－9
（1998. 12. 4）

赤外線通信を用いた頑健なモバイルアドホックネットワーク構築手法

片桐秀樹† 河口信夫†
外山勝彦† 稲垣康善†

我々はこれまでに，ユーザが携帯端末を持ち寄るだけで端末がお互いを認識し自律的にモバイルアドホックネットワークを構築する手法を提案してきた．しかし，従来の手法ではネットワーク構築中のトポロジの変更や通信エラーなどに対する考慮が十分ではなかった．本稿では，頑健なアドホックネットワークを構築する手法を提案する．本手法では，従来の手法に加えて，各端末が隣接ノードを発見する手続きを定期的に行い，ネットワーク構築中の動的なトポロジ変化に対応する．また，様々なイベントに対応することによって，通信エラーからの復帰処理を行う．

Construction of Robust Mobile Ad-Hoc Network using Infrared Communicaion

HIDEKI KATAGIRI,† NOBUO KAWAGUCHI,† KATSUHIKO TOYAMA† and YASUYOSHI INAGAKI†

We have proposed an autonomous communication protocol for an ad-hoc network. But we have supposed no dynamic change of the the network topology while mobile hosts construct a network, and no communication errors. In this paper, we propose a new approach that each mobile hosts autonomously deal with dyanmic changes of network topology by doing periodic discovery while constructing a network. It enables mobile hosts to recover from communication errors.

1. はじめに

ノートパソコンやハンドヘルドPCに代表される小型で高機能な携帯端末が普及し，いつでもどこでも必要な時に様々な情報の利用が可能になりつつある．より多くの人々が携帯端末を持つようになると，会議などの多くの人々が集まる場は同時に多数の携帯端末が集まる場となる．このような場で，ユーザが持ち寄った複数の携帯端末間でネットワークを構築し，保存されている情報の交換や共有を行いたい要求が最近高まりつつある．携帯端末間で直接，名刺交換やスケジュール調整，議事録の共有やメモの交換，資料の配布等を行うことで，より効率的な共同作業の支援が可能になると考えられる．しかし，携帯端末間の通信は，面倒な作業や，特別な無線通信機器[4)5)]を必要するため，実際にはほとんど行われていない．

この問題を解決するために，我々はすでに，いつでもどこでも手軽に，携帯端末間の通信を可能にする自律分散通信プロトコルを提案した[1)2)]．各携帯端末は特別な設定を必要とせず，自律的に1対1通信を繰り返して，他の端末やネットワークに関する情報を共有し，一時的なネットワークを構築する．本研究では，通信媒体として，すでに多くの携帯端末で利用可能な赤外線通信[8)]を対象としている．

このように端末が自律的に周囲の端末を認識し，一時的に構築するネットワークは，アドホックネットワークと呼ばれる．アドホックネットワークの構築においては，(1) ネットワーク中に存在するループへの対応，(2) 複数端末によるネットワーク構築の開始，(3) 直接通信できない端末間におけるルーティング，(4) 端末の移動に伴う動的なネットワークトポロジ変化への対応，を考慮する必要がある．(1) ～ (3) の問題に対しては従来のネットワーク構築手法で対応済である．しかし，ネットワーク構築中の動的なトポロジの変化はないと仮定していたため，(4) の問題には完全には対応できていなかった．

本稿では，従来の手法を拡張し，ネットワークトポロジの動的な変更や通信エラーなど様々なイベントに対応

† 名古屋大学大学院工学研究科計算理工学専攻
Department of Computational Science and Engineering, Graduate School of Engineering, Nagoya University

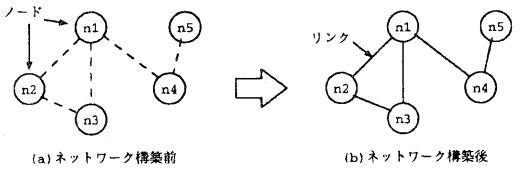

図1 ノードとリンク
Fig. 1 Nodes and links

する，頑健なアドホックネットワークを構築する手法を提案する．本手法では，各端末がネットワークトポロジの変化を監視するために，通信を行わない間，定期的に隣接する端末の発見手続きを繰り返す．隣接する端末と通信不可になった場合には，その通信リンクをネットワーク情報から消去し，他の端末に報告する．また，新たに隣接する端末を発見した場合には，その時点で通信を行い，従来の手法よりも効率良く通信リンクの存在を他の端末に伝える．

2章では，従来の手法を用いてネットワーク構築を行う際の問題点とその解決手法について述べる．3章では，拡張したアドホックネットワーク構築の手法について述べる．4章では，提案する手法の検証を行うためのTCP/IP上のエミュレータ環境について述べる．

2. 問題解決のアプローチ

アドホックネットワークを構築する際の問題を簡単化するため，従来の手法ではネットワーク構築中の動的なトポロジの変化や，通信エラーはないと仮定していた．この仮定を無くした場合，各端末は予期しないイベントにより，デッドロック状態に陥ることがある．本章では，この問題点について述べ，次に解決のための基本方針を述べる．

以下では，携帯端末をノードと呼び，固有な識別子n_iを持つ円で表す．ノード間の通信路をリンクと呼び，直接通信可能なリンクを点線で，各ノードが認識しているリンクを実線で表す(図1)．ネットワーク構築手続きを開始するノードを探索開始ノードと呼び，あるノードに対して直接通信可能なノードを隣接ノードと呼ぶ．

2.1 従来のネットワーク構築手法の問題点

従来の手法[3]では，各端末は同じアルゴリズムに従い，他の端末や通信リンクに関する情報(以下ネットワーク情報)を交換し，他の端末と協調し，アドホックネットワークを構築する(図1)．各ノードが従うアルゴリズムは，分散アルゴリズムのDiffusing Computation[7]を応用している．アドホックネットワークの構築は，以下に示す3ステップで行われる．各ノードは，隣接ノードとネットワーク情報を送受信することにより，図2に示す状態を遷移する．

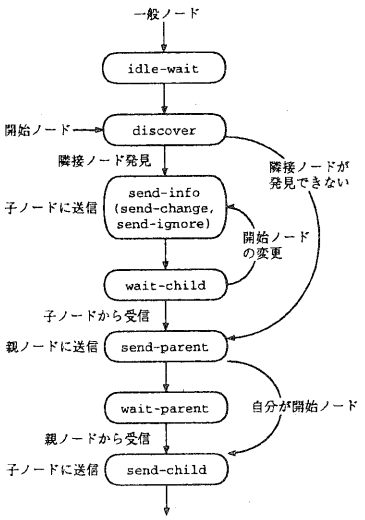

図2 従来のネットワーク構築手法の概要
Fig. 2 Former method of an ad-hoc network construction

Step1 構築するネットワークを木構造，探索開始ノードを木構造の根とみなす．探索開始ノードから木構造の葉に相当する末端のノードへ向かって，各ノードが隣接ノードの発見手続きを行い，発見したノードにネットワーク情報の送信を繰り返し行う(discover → send-info → wait-child)．

Step2 Step1で用いた経路を逆に，末端ノードから探索開始ノードへネットワーク情報を収集する．これにより，探索開始ノードは全てのネットワーク情報を獲得する(send-parent → wait-parent)．

Step3 全てのノードが同じネットワーク情報を持つように，探索開始ノードから末端ノードまでStep1で用いた経路をたどって，各ノードがネットワーク情報の送信を繰り返し行う(send-child)．

ここで，Step2とStep3でネットワークトポロジが変化した場合，通信リンクが切断されたノード間ではネットワーク情報の交換が行われないため，これらのノードはデッドロックに陥る．以下でネットワークトポロジが動的に変化したために，ノードが陥るデッドロックの典型的な例について，図3を用いて示す．ノードn_pを探索開始ノードとし，このノードからネットワークの探索が開始されたと仮定する．

(例1) ノードn_iは，Step2において子ノードn_jから送信されるネットワーク情報を待機する．ノードn_iが移動して$n_i - n_j$間の通信リンクが切断さ

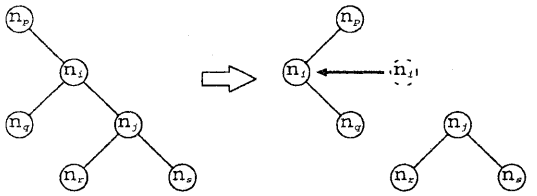

図3 ノードの移動に伴うリンクの切断
Fig. 3 Link disconnection

れた場合(図3(b)), ノードn_jは親ノードn_iにネットワーク情報の送信を試みるが, リンクが切断されているため, 通信エラーとなり, n_iにネットワーク情報を送信できない. また, ノードn_iは子ノードn_jから送信されるネットワーク情報を待ち続ける. その結果, 探索開始ノードn_pにネットワーク情報を送信することができず, Step2が終了しない.

(例2) ノードn_jは, Step3において親ノードn_iから送信されるネットワーク情報を待機する. ノードn_iが移動して$n_i - n_j$間の通信リンクが切断された場合(3(b)), ノードn_iは子ノードn_jにネットワーク情報の送信を試みるが, 通信リンクが切断されているため, 通信エラーとなり, n_jにネットワーク情報を送信できない. また, n_jは親ノードn_iから送信されるネットワーク情報を待ち続ける. その結果n_j, n_r, n_sは全体のネットワーク情報を獲得することができない.

以上のように, ネットワーク構築中にトポロジが変化した場合, 従来のアルゴリズムでは, 目的のノードとネットワーク情報の交換を行うことができない. したがって, ノードはアルゴリズムを終了することができず, 全体のネットワーク構築も完了しない.

2.2 デッドロックの回避

前節で述べたノードがデッドロックに陥る問題は, 各ノードがネットワーク情報の待機中に, 動的なトポロジの変化を監視することで解決可能である. したがって, 各ノードがネットワーク情報の待機中も, 隣接するノードの発見手続きを行うようにする. この解決手法を用いたノードの基本動作例について, ネットワーク構築中にトポロジが変化し, 隣接ノードと通信不可になった場合と, 新たに通信可能なノードが現れた場合を示す.

- ノードn_iが子ノードn_j, n_qからのネットワーク情報受信待機状態で, ノード$n_i - n_j$間のリンクが切断された場合(図3)
  (1) ノードn_i, n_jは, お互いに関する情報をそれぞれが管理するネットワーク情報から削除する.
  (2) ノードn_iは, 残りの子ノードn_qからネットワーク情報を待機する. n_qからネットワーク情報を受信したら, 親ノードn_pにn_i, n_jのリンク情報を除いたネットワーク情報を送信する. そして, n_pからのネットワーク情報の待機中に, 子ノードn_qに切断したn_i, n_j間のリンク情報を送信する.
  (3) ノードn_jは, 親ノードがいなくなってしまったので, 以後は自ノードが木構造の根に相当する探索開始ノードとして振舞う. 子ノードn_r, n_sからネットワーク情報を受信するまで待機し, 受信したら切断した$n_i - n_j$間のリンク情報を子ノードに送信して, n_j, n_r, n_sから構成されるネットワークを構築する.
- ノードn_iが子ノードn_qからのネットワーク情報受信待機状態で, 新たに隣接ノードn_jを発見した場合(図4)
  (1) ノードn_iは, ノードn_jに対してネットワーク情報を送信し, 子ノードとして登録する.
  (2) ノードn_jは, n_iを親ノードとし, 隣接ノードの発見手続きを行う. 新たな隣接ノードがないので, 親ノードn_iに対して, ネットワーク情報を送信する.
  (3) n_iは, n_j, n_qからネットワーク情報を受信したら, 同様に親ノードn_pにノードn_jを含むネットワーク情報を送信する. そしてn_pからのネットワーク受信待機状態の時に, 子ノードn_qに新たに加わったネットワーク情報を送信する.

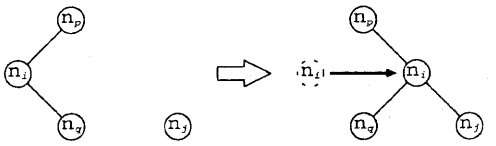

図4 ノードの移動に伴う新たな隣接ノードの発見
Fig. 4 Discovery of a new adjacent node

以上のように, 各ノードが隣接するノードの発見手続きを待機中も行うことによって, 動的なトポロジの変化があった場合でも, デッドロック状態に陥ることはない. そして, 互いに通信可能なノードから構成されるアドホックネットワークを構築する.

2.3 イベントに基づくノード動作の記述

従来の手法では，各ノードは隣接ノードとのネットワーク情報の送受信により，状態を遷移させ，ネットワーク構築を行っていた．前節で述べたトポロジの変化に伴うリンクの切断や新たな接続，通信エラーへの対応を考慮した場合，各ノードは他のノードと非同期に動作する必要がある．したがって，さまざまなイベントへの対応を状態遷移図に組み込む必要があるが，その記述が難しくなってしまう．

そこで，本稿では，各ノードは他のノードとの通信やネットワークトポロジの変更等をイベントとしてまとめて取り扱う．そして，受け取ったイベントによって，状態を変更する．ノードの動作をイベント中心の記述に変更することにより，任意の状態において新たなイベントが追加されても，その解決の見通しが立ちやすくなる．

3. 頑健なアドホックネットワーク構築手法

本章では，各ノードの動作をイベントによって決定する頑健なアドホックネットワーク構築手法の提案をする．

3.1 前　　提

ノードとリンク，およびノード間の通信に関する前提を以下に挙げる．

(1) 各ノードは固有なIDを持つ
(2) 初期状態では各ノードは他のノードや通信リンクに関する情報は持たない
(3) 各ノードは自発的に直接通信可能なノードと，通信リンクの存在を発見できる
(4) リンクは双方向に通信可能である
(5) 各ノード間は，同時には高々一つのノード通信を行うことができる

赤外線通信の規格であるIrDA[8]は，HDLC手順に基づく半二重の1対1通信であり，前提(5)はこの性質に基づいている．

3.2 準　　備

各ノードn_iは，探索開始ノードを示す変数I_iと，隣接ノードとの通信履歴の管理を行い，他のノードに関する情報や送受信した隣接リストを保持するための情報テーブル$T_i = [t_i^j, \cdots, t_i^k]$を持つ．$t_i^j = (n_j, in_i^j, out_i^j, Rel_i^j, AL_i^j)$はノード$n_j$に対応する5項組である．$n_j$はノードの固有IDであり，$in_i^j, out_i^j$はノード$n_j$と隣接リストを受信，送信したかどうかを示すフラグである．Rel_i^jはn_jとの関係を表し，{M,N,P,D,C,I,A}のいずれかの値をとる．それぞれの記号の意味を表1に示す．AL_i^jはノードn_jから受け取った隣接リストを表す．隣接リストとは，ノード間の接続関係を示すものであり，各ノードの固有IDの対である隣接ペアの集合である．図1の場合，隣接リストは$\{(n_1,n_2),(n_1,n_3),(n_1,n_4),(n_2,n_3),(n_4,n_5)\}$となる．各ノードがネットワーク全体のトポロジを示す隣接リストを獲得することにより，直接通信できないノードの存在を認識し，ルーティングの計算を行うことができる．

表1 隣接ノードとの関係
Table 1 Relations between neighbor nodes

変数	意味
M	自ノード
N	一般ノード
P	親ノード
D	子ノード
C	開始ノード・親ノードの変更通知を行うノード
I	隣接リストを送受信しないノード
A	通信不可になったノード

表2 情報テーブルの例(図1 n_3の場合)
Table 2 Example of an information table

ID	in	out	Rel	AL
n_3	f	f	M	$[(n_3,n_4)]$
n_1	t	f	P	$[(n_1,n_2),(n_1,n_3)]$
n_2	t	t	I	$[(n_2,n_3)]$
n_4	f	t	D	[]

各ノードは，隣接リストをやりとりするために，以下のメッセージを送受信する．ここでn_iはメッセージの送信側のノードのIDを示す．

IDandAL(n_i, I_i, AL_i)
ノードn_iの探索開始ノード変数I_i，隣接リストAL_iを送る

AL(n_i, AL_i)
ノードn_iの持つ隣接リストAL_iを送る

Ignore(n_i)
ノードn_iと間で隣接リストを送受信を行わない

Change(n_i, I_i, AL_i)
親ノードをn_iに，探索開始ノード変数をI_iに変更するように指示し，隣接リストAL_iを送る

AddAL$(\{n_x, n_y, \cdots\}, AL)$
ノードの集合$\{n_x, n_y, \cdots\}$に，追加する隣接リストALを送る

DelAL$(\{n_x, n_y, \cdots\}, AL)$
ノードの集合$\{n_x, n_y, \cdots\}$に，削除する隣接リストALを送る

3.3 ネットワーク構築アルゴリズム

各ノードは，表3に示すイベントテーブルおよび表4の関数表に従って動作する．基本的な動作としては従来の手法と同じアルゴリズムに従うが，idle-wait,wait-

info,wait-parent の各状態で，定期的に隣接するノードの発見手続きを行い，トポロジの動的な変更に対応する．

各ノードは，イベントテーブル中で以下の関数を利用する．

- Detect()
  発見した隣接ノードの ID をリストとして返す
- Send(msg)
  メッセージ msg を送信する．宛先は，メッセージ内に含まれる

また，各ノードは以下のようなイベントを受け取る．

- Recv(msg)
  メッセージ msg の受信を示すイベント☆．
- x is new node
  ノード n_x が新たな隣接ノードとして発見されたことを示すイベント
- x is away node
  ノード n_x が隣接ノードではなくなったことを示すイベント

初期状態では，ノード $n_\alpha(\alpha = i, j, \cdots)$ の探索開始ノード変数は $I_\alpha = \epsilon$ であり，情報テーブルは $T_\alpha = [(n_\alpha, f, f, M, [])]$ である．

全てのノードは，idle-wait 状態からスタートする．各ノードは，表3中のイベント (Event) を受け取ると，現在の状態 (CurrentState) とテーブル中の変数や探索開始ノード変数の値によって，Action(s) に記述された動作を行い，次の状態 (NextState) へ遷移する．NextState 中の関数は表4に従って，次状態を返す．

以下では，章で挙げたアドホックネットワーク構築における問題点 (1)，(2)，(4) を解決する過程を以下で述べる．また，テーブルとの対応を ⋆ で示す．

- ネットワーク内にループが存在する場合
  複数のノードから同じ探索開始ノード変数を含む IDandAL メッセージを受信 *1 するノードが存在する．この場合，ネットワークを木構造とみなすために，後から IDandAL メッセージを送信してきたノードに対し，ネットワーク構築中はネットワーク情報の交換を行わない旨の Ignore メッセージを送信 *2 する．
- 探索開始ノードが複数の場合
  異なる探索開始ノード変数を含む IDandAL メッセージを複数のノードから受信 *3 するノードが存在する．この場合，このノードは新たな探索開始ノードとなり，IDandAL メッセージを送信したノードに対して，親ノードと子ノードの役割を入れ換えるための Change メッセージを送信 *4 する．この結果，ネットワーク中に探索開始ノードが一つにまとめられ，効率良くネットワーク情報を獲得できる．
- 待機中に隣接ノードがいなくなった場合
  定期的な隣接ノードの発見手続きの結果，あるノード n_i に対して，隣接するノード n_j が通信不可になった場合 *5，まず隣接リストから隣接ペア (n_i, n_j) を削除する．テーブル中に，$Rel_i^k = \mathrm{I}$ であるノード n_k が存在すれば，このノードを親ノードにして，ネットワーク構築を行う．そのようなノード n_k が存在しなければ，自分を探索開始ノードとみなして，ネットワークを構築する．
- 待機中に新たな隣接ノードが現れた場合
  定期的な隣接ノードの発見手続きの結果，あるノード n_i に対して，新たな隣接ノード n_j が現れた場合 *6，まず隣接ペア (n_i, n_j) を隣接リストに付け加える．そして，n_j に対して IDandAL メッセージを送信する．これによって，n_i は n_j を子ノードとして取り扱い，ネットワーク情報の送受信を n_j との間で行う．図5にノード n_i, n_j の状態遷移と交換するメッセージを示す．
- 目的のノードと通信できない場合
  定期的な隣接ノードの発見手続きの結果，直接通信可能であるはずのノードとなんらかの原因で通信できない場合には，このノードとの通信リンクが切断されたと見なして，隣接ノードがいなくなった場合と同じように動作する．これにより，送信側のノードは，メッセージの再送を繰り返さず，他の動作を行う．

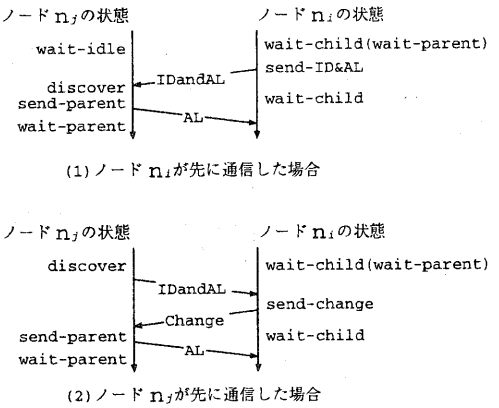

図5 ノード n_i, n_j の状態遷移と交換するメッセージ
Fig. 5 State of nodes n_i, n_j and messages

☆ 表3中では受信したメッセージ msg のみを記す

表3 イベントテーブル
Table 3 Event table

Event	Current State	条件	Action(s)	Next State
IDandAL(n_x, I_x, AL_x)	idle-wait	(なし)	add(n_x, P, AL_x),$I_i := I_x$ Discover()	func-discover()
	send-info	$I_i = I_x$ ⋆1	$Rel_i^x = I$	func-info()
		$I_i \neq I_x$ ⋆3	$Rel_i^x = C$	func-info()
	wait-child	(なし)	add(n_x, C, AL_x)	func-info()
	wait-parent	(なし)	add(n_x, C, AL_x)	func-info()
	normal	(なし)	add(n_x, P, AL_x)	func-parent()
Change(n_x, I_x, AL_x)	wait-child	$I_i = n_i$	$Rel^x = P, I_i = I_x$	recv-child()
		$I_i \neq n_i$	Rel^y = C where $Rel^y = P$ Rel^x = P,$I_i = I_x$	func-info()
Ignore(n_x)	wait-child	(なし)	Rel^x = I	recv-child
Recv(AL(n_x, AL_x))	wait-child	$Rel^x = D$	$AL_i^x += AL^x$	recv-child
		$Rel^x = I$	$Rel^x = D, AL_i^x += AL^x$	recv-child
	wait-parent	$Rel^x = P$	$AL_i^x += AL^x$	func-child()
		$Rel^x = I$	Rel^x = D,$AL_i^x += AL^x$	wait-parent
	normal	Rel^x = I	Send(AL($n_x, \cup_j AL_i^j$)) Send(AddAL(n_j, AL_x))	normal
AddAL(n_x, AL_x)	normal	(なし)	Send(AddAL(AL_x))	normal
DelAL(n_x, AL_x)	wait-parent	(なし)	Send(DelAL(AL_x))	wait-parent
	normal	(なし)	Send(DelAL(AL_x))	normal
x is new node ⋆6	idle-wait discover,wait-child wait-parent,normal	(なし)	add($n_x, N, []$) $AL_i^i += (n_i, n_x)$	func-info()
x is away node ⋆5	wait-child	$Rel_i^x = P$ and $\exists y$ where $Rel_i^y = I$	del(n_i, n_x) Rel_i^x = A,$Rel_i^y = P$	recv-child()
		$Rel_i^x = P$ and $\nexists y$ where $Rel_i^y = I$	del(n_i, n_x) $Rel_i^x = A, I_i = n_i$	wait-child
		$Rel_i^x \neq P$	del(n_i, n_x)	wait-child
	wait-parent	$Rel_i^x = P$ and $\exists y$ where $Rel_i^y = I$	del(n_i, n_x) Rel_i^x = A,$Rel_i^y = P$	func-parent()
		$Rel_i^x = P$ and $\nexists y$ where $Rel_i^y = I$	del(n_i, n_x) $Rel_i^x = A, I_i = n_i$	func-child()
		$Rel_i^x \neq P$	del(n_i, n_x)	wait-parent
	normal	(なし)	del(n_i, n_x) Send(DelAL(n_i, n_x))	normal

3.4 ルーティングテーブルとデータの配送

構築したネットワークにおいて，リンクがないノード間の通信は他のノードを中継して行う．したがって，各ノードはどのノードを中継に用いるかを決定しなければならない．本手法では以下のように行う．

まず各ノードは，獲得した隣接リストから，他の各ノードへの最小パスを全域木の最小パスを求めるアルゴリズムによって計算し，ルーティングテーブルを作成する．あるノードから直接通信できないノードへデータを送信する場合は，ルーティングテーブルを参照して中継先を決定し，目的のノードにデータが到達するまで中継を繰り返す．また，複数のノードへ同じデータを送信する場合には，中継先毎にデータを送信し，疑似的なマルチキャストを実現する．

4. プロトコルエミュレータ

提案したネットワーク構築手法の動作確認と，実装におけるプログラミングのデバッグ等を容易にするために，赤外線通信をエミュレートする環境を構築した．図6にエミュレーション環境上でのプロトコルの動作の様子を示す．左上のウィンドウがエミュレーションサーバであり，各ノードの通信を管理する．画面上には，各ノードの接続，及び通信の様子を表示する．サーバ上で，ノード間の接続状況を変更することにより，動的なネットワークトポロジの変更をシミュレートすることができる．また，複数の小さなウィンドウが各ノードを実現するプログラムであり，情報テーブルや通信の様子を表示している．

各ノードを実現しているプログラムは，ソケット通信を用いて独立にエミュレーションサーバと通信を行って

表4 次状態を決定する関数
Table 4 Next state function table

Function	条件	Action(s)	Next State
func-info()	(なし)	foreach $n_j \in \mathrm{ID}(T_i)$ and out_i^j = f if Rel_i^j = N then Send(IDandAL($n_i, n_j, I_i, \cup AL_i$)) Rel_i^j = D else if Rel_i^j = C then Send(Change(n_i, n_l, I_i, AL_i)) ⋆4 Rel_i^j = D else if Rel_i^j = I then Send(Ignore(n_i, n_j)) ⋆2 out_i^j = t	wait-child
fucn-parent()	$\exists n_j$ where Rel_i^j = P	Send(AL($n_i, n_j, \cup_{x \neq j} \mathrm{AL}_i^x$))	wait-parent
	$\nexists n_j$ where Rel_i^j = P	(なし)	send-child
func-child()	(なし)	foreach n_j where Rel_i^j = D Send(AL($n_i, n_j, \cup_{x \neq i,j} AL_i^x$))	normal
func-recv()	$\exists n_j$ where in_i^j = f and Rel_i^j = D	(なし)	wait-child
	$\forall n_j$ where in_i^j = t and Rel_i^j = D, $I_i = n_i$	(なし)	send-child
	$\forall n_j$ where in_i^j = t and Rel_i^j = D, $I_i \neq n_i$ out_i^k = f where Rel_i^k = P	(なし)	send-parent
	$\forall n_j$ where in_i^j = t and Rel_i^j = D, $I_i \neq n_i$ out_i^k = t where Rel_i^k = P	(なし)	wait-parent
func-discover()	x is new node	add($n_x, N, []$), AL_i+ = (n_i, n_x)	send-info
	new node is none and $I_i = n_i$	(なし)	normal
	new node is none and $I_i \neq n_i$	(なし)	send-parent

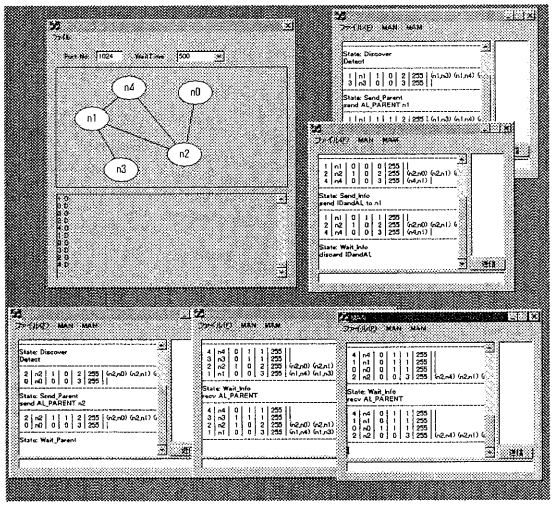

図6 エミュレーション環境での実行画面
Fig. 6 Emulation environment

いる．したがって，ソケット通信の接続先を赤外線通信のインタフェースに変更するだけで，実際のシステムを実現できる．

現在，このエミュレーション環境を用いて，様々な状況を想定して，提案した通信プロトコルの動作確認を行っている．

5. ま と め

本稿では，我々がこれまでに提案した手法では対応できていなかった，ネットワーク構築中の動的なトポロジの変更に対応するための頑健なアドホックネットワーク構築手法を提案した．定期的に直接通信可能な端末の発見手続きを行い，非同期なイベントに対応することで，ネットワーク構築中の動的なトポロジ変化に対して柔軟に動作を行う．また，本手法の動作を確認するためのエミュレーション環境を構築した．

本手法により，ユーザが端末を持ち寄るだけで自律的にアドホックネットワークの構築が行われ，ネットワークに参加する任意の端末間で通信が可能となる．また，ユーザのアドホックなコミュニケーションを支援できる．

今後の課題として，プロトコルの妥当性の検証やメッセージ交換の効率性の向上，端末数の増加に伴うスケーラビリティの問題が挙げられる．また，データを送信す

る場合，他の端末を中継して行うので，メッセージの暗号化のようなセキュリティの確保の必要がある．

参 考 文 献

1) 片桐 秀樹, 河口 信夫, 外山 勝彦, 稲垣 康善：モーバイル環境下における赤外線を用いた自律分散通信プロトコル, DiCoMo, pp67-72, 1997.
2) 河口 信夫，片桐 秀樹，内柴 道浩，外山 勝彦，稲垣 康善：モーバイル環境下の自律分散通信の実現とその応用，情報処理学会,DiCoMo98, pp.619-626, 1998.
3) 片桐 秀樹, 河口 信夫, 外山 勝彦, 稲垣 康善：モバイルアドホックネットワーク構築のための分散アルゴリズムの提案とその実現, 情報処理学会第 57 回全国大会, vol 3, pp.558-559, 1998.
4) 多鹿 陽介, 岩村 和昭, 池上 史彦, 中村 誠: 携帯情報機器の通信に適した自律無線ネットワーク Wireless DAN の提案, 信学技報, IN 94-161,1995.
5) 倉島顕尚, 市村 重博, 田頭 繁, 前野 和俊, 武次 将徳, 永田 善紀: 集まったその場での協同作業を支援とするモバイルグループウェアシステム「なかよし」,Dicomo(マルチメディア, 分散, 協調とモーバイル), pp233-238,1997
6) 中村 眞, 藤井 章博, 根元 義章: 移動分散環境下でのネットワークトポロジ把握のためのプロトコル, 情処研報, DPS 66-12, 1994.
7) Michel Raynal:Distributed Algorithms and Protocols, Wiley, 1988.
8) Infrared Data Association:Serial Infrared Link Access Protocol,version 1.1, 1996.