ヒューマンインタフェース 44−23
（1992. 9. 11）

無線通信におけるマルチキャストを用いた同報応答機構MACSの設計と実装

岩沢 透　矢向 高弘　安西 祐一郎
慶應義塾大学理工学部

無線通信を用いて Contract Net を実現する場合には、同報送信に対する応答機構が必要である。しかし、同報送信に対して複数の応答を受信するためには多大な時間を要する。本研究では、一部の局だけに応答をさせるマルチキャストを用いることにより、応答数の少ない同報応答機構 MACS(Mechanism of ACknowledgement using multicaSt) を設計・実装した。MACS は、応答受信中も通信経路を占有しないので、他の通信が疎外されることがない。さらに、同時に複数の Contract Net を起動し、応答を並行に受信することも可能である。

MACS: Response mechanism using multicast in a radio network

Toru Iwasawa　Takahiro Yakoh　Yuichiro Anzai
Faculty of Science and Technology, Keio University

When we want to realize Contract Net protocol, a response mechanism for broadcast is needed. But it takes so much time to receive many acknowledgements. In this paper, we have designed and implemented a response mechanism called MACS(Mechanism of ACknowledge using multicaSt) by using multicast announcement. Using MACS, communication channel is never occupied even if responses are sending. Moreover, multi Contract Net protocols can be controlled concurrently without occuring colligion of many acknowledgements.

1 はじめに

自律移動ロボットを用いたロボットどうし、あるいは人間・ロボット間の協調作業の研究は、人間-コンピュータ間の新たなインタフェースの研究として近年注目されている [1]。

自律移動ロボットを用いた協調作業を実現する上で重要となる項目の一つに、ロボットへのタスク割り当ての方法がある。タスク割り当ての方法の一つとして Contract Net [9] がある。Contract Net を応用したタスク割り当ての方法は、ロボットへのタスク割り当てをする際に有効であると考えられる。Contract Net は、タスクを割り当てるノードが周囲にタスクの内容を宣言し、返ってきた応答の中から実際にタスクを割り当てる相手を決定する方法を用いる。従って、Contract Net を無線通信が可能な移動ロボットを用いて実現するためには、同報送信に対する応答機構が必要になる。

無線パケット通信を用いて同報送信に応答する機構を実現する際には、応答の送信方法が重要である。なぜなら、同報送信を受けとった側が一斉に応答を送信すると応答パケットの衝突が起こってしまうからである。応答パケットの衝突を防ぐために、応答を時間的にずらして送信する方式がいくつか、提案されている [2] [3]。しかし、多数の応答の受信には時間がかかり、その間他の送信は応答送信の妨げとなるので抑えられる。

そこで本研究では、応答が衝突を起こさず、応答送信が行われている間も他の送信ができる応答機構 MACS の設計、実装を行う。MACS は、同報送信に対する応答が行われている間でも他の同報送信を行い、それに対して衝突を起こさずに応答を受信することができる。

以下の章では、まず Contract Net について少し説明を加え、次に MACS の設計について述べる。次に MACS の実装について述べ、実装した MACS を用いた評価の結果について述べる。

2 Contract Net

この章では、ロボット間のタスク割り当てに有効であると思われる Contract Net について述べる。

Contract Net は、ノード群間でタスク割り当てを行うためのプロトコルであり、ノード間のタスク割り当ては以下のようにして行われる。

- タスク割り当てを行うノード（マネージャ）がそのタスクを宣伝する (Task Announcement)。
- その宣伝を受信したノードがそのタスクの要求に応じられる時はそれに対して入札する (Bidding)。
- マネージャは入札の情報に基づいてタスクを割り当てるノードを決定し、タスクを割り当てる契約を結ぶ (Award)。

Contract Net を用いてロボットどうしが協調作業をする応用例としては、以下のものがある。

- センシング能力はあるが、得られたセンサのデータを処理して自分の行動の手がかりとすることができないロボットが処理能力の高いロボットを探し、そのロボットに処理を代行させる作業
- 2 台のロボットが同じ場所へ荷物を届けようとしている時、どちらかのロボットがもう一台のロボットの荷物を受けとり目的地へ運ぶ作業

Contract Net において、タスクの存在を同報送信 (アナウンス) するロボット (Manager) は特別な場合を除き、すべての相手から応答を受け取る必要はない。例として、あるロボットが同じ方向へ向かうロボットに荷物を届けてもらう例を考えてみる。この場合、もし周囲に同じ方向へ行くロボットが複数いるのなら、周囲にいるすべてのロボットから応答を得ることができなくても目的のロボットから応答を受けとることは可能である。

3 MACS の設計

ここでは、前節で述べた Contract Net の特徴をふまえて MACS の設計を行う。MACS の設計の際には、次の仮定をしている。

- 無線通信の送信パワーは、すべてのロボットで一定である。従って同報送信を受信したロボットは、送信元のロボットに対しても送信可能である
- ロボットの動作する環境は、オフィス内のような比較的範囲の限定された場所とし、ロボットが孤立するような状況は考えないものとする

3.1 応答の返送方法

無線通信において、TDMA（時分割多重方式）は衝突を起こさずにパケットを送信できるため、伝送路の使用効率がよくなるという特徴を持つ。これを利用して、伝送路の効率をさらに高めようとさまざまな研究が行われている。[5]～[8]。

本研究では、応答の送信には TDMA を用いる。応答の送信方法は次の通りである。まず同報送信を同

期信号とし、応答を送信するロボットが同期をとる。そして、応答を送信する順に各ロボットに重複のないタイムスロットを割り当てる。そして、応答をするロボットは自分が応答できる時刻まで待機した後、応答を返送する。各ロボットは1個の同報送信に対してのみ、応答送信待ちができる。

ここで、もし応答が連続して送信されると、同報送信に対し応答が送信されている間は他のロボットは応答送信の妨げとなるので通信をすることができない。

そこでMACSは、タイムスロットの割り当てに関して、空スロットを割り当てる方法を導入する。これは、応答送信の間に空のタイムスロット (Empty slot) を用意するもので (図1参照)、応答以外の送信をする場合はこのタイムスロットを利用するものとする。し

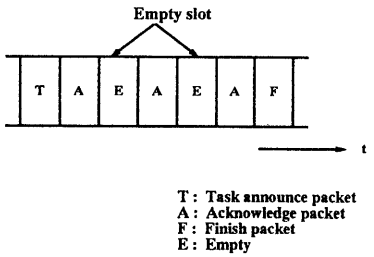

図 1: 空スロットの割り当て

かし、空スロットの導入だけでは空のタイムスロットを利用して別の応答送信を要する同報送信を行うことはできない。なぜなら、新たな同報送信に対する応答用のタイムスロットを作る余地がないからである。そこで、今まで応答の受信待ちをしていたロボットのタイムスロットを移動し、新たなメッセージに対する応答用スロットを作る方法を提案する。新たな応答用スロットは、既に応答待ちになっているロボットが送信を遅らせることによって作成される (図2参照)。具体的なタイムスロットの割り当てについては3.4節で詳しく述べる。

また、応答送信の順番は各ロボットに固有のid番号を与えておき、その番号の小さいロボットから順に応答を送信することを基本とする。

3.2 同報送信の形態

Contract Net において、タスクの存在を同報送信 (アナウンス) するロボット (Manager) は特別な場合を除き、自分から送信可能な範囲にいるすべての相手から応答を受け取る必要はない。特別な場合というのは、自分から直接送信可能な相手を知りたい場合など

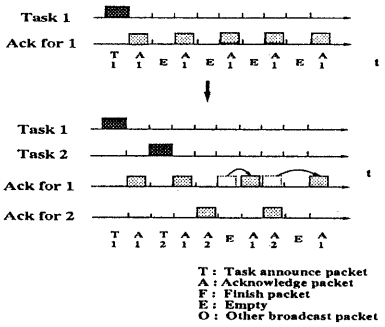

図 2: タイムスロットの移動

である。従って、同報送信の手段としてはすべてのロボットを対象としたブロードキャストよりは、あるグループを対象とするマルチキャストで十分であると考えられる。そこで、MACSの同報送信にはマルチキャストを用いることにした。マルチキャストの方法を次に示す。

まず、応答の送信方法であるが、常にid番号の小さいロボットから順に何台か応答を送信させる方法を用いると、id番号の小さいロボットほど応答をする機会が多くなり、不公平である。そこでMACSでは同報送信の際、メッセージのヘッダ部にhead idというidを付加しておき、head idで指定されたロボットから順に応答を送信する。head idの決定法はメッセージの送信方法で述べる。

マルチキャストの宛先は、以下に示す方法でhead idとタイムアウトするまでの時間から受信側が判断する。

- 同報送信を行うロボットは、head idとタイムアウトするまでの時間をメッセージに付加する。
- 同報送信を受信したロボットは、head idとタイムアウトの時間から、同報送信したロボットがタイムアウトする前に、自分が応答を送信するタイムスロットが回ってくるかどうかを判別できる。
- 同報送信を行ったロボットがタイムアウトする前に自分が応答をするためのタイムスロットが回ってくるなら、そのロボットはマルチキャストの指定先に含まれていると判断する。

3.3 同報送信の方法

同報送信を行う際の留意点として、ロボットが通信範囲内にいないことがあるため、指定したグループ内

のロボットの数だけ応答が返ってくるとは限らないことがある。マルチキャストの方法（宛先など）を設計する際には、この点を考慮する必要がある。

そこで、MACSでは、同報送信を行う際、必要とする応答の個数とタイムアウトして応答の受信を終了するまでの時間をメッセージに付加する。そして同報送信を受信したロボットが同報送信の内容から自分が応答をする必要があるのかどうかを判別できるようにする。この方法を用いると、応答の個数の必要性 (必要なだけの数は欲しい、足りなくても構わない *etc.*) に応じて、タイムアウトの時間をユーザ（アプリケーションを書く人）が調整できる。すなわち、どうしても指定した数の応答が必要な場合はタイムアウトするまでの時間を長くし、足りなくても構わない場合は短くするというようにユーザが指定できるようにする。

以上のことを考慮し、同報送信にあたって必要な情報は、

- head id とタイムアウトするまでの時間
- 自分が送信したメッセージと他のロボットが送信したメッセージを区別するためのメッセージ id
- 送信するメッセージの内容

ということにした。

head id の決定は、同報送信するロボットが内部で行う。head id の決定法は、他に応答受信中のロボットが存在しなければランダムに決定する。応答受信中のロボットが存在する場合は、既に他の同報送信に対して応答を送信待ちであるロボットを除き、残ったロボットが最も多くマルチキャストの宛先に含まれるように head id を決定する。

次に、応答送受信の例を示す。例として、head id を 2、必要とする応答の数を 3、タイムアウトするまでの時間を 10、スロットの 1 周期にかかる時間（スロットサイクル）を 2 とした時の応答送受信の動作を図3に示す。この場合、タイムアウトするまでに応答返送が可能なロボット数は、タイムアウトまでの時間を t_{out}、スロットサイクルの時間を sl とすると

$$t_{out}/sl = 10/2 = 5$$

となる。故に、タイムアウトするまでに応答スロットが回ってくるロボットは 5 台である。従って、id が head id = 2 のロボットから 6 のロボットまで応答のスロットが回ってくるので送信を待つ必要がある。それ故、マルチキャストの宛先は、2,3,4,5,6 であると考えることができる。そして、必要となる応答の数は 3 個なので、2,3,4 は確実に送信するが、5,6 は、送信しない可能性もある。なお、メッセージ id は応答を受信する際に、その応答が自分が行った同報送信に対して返ってきたものかどうかを判別するために用いる。

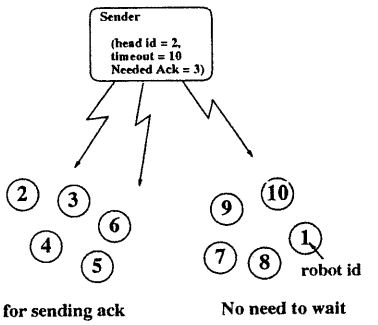

図 3: 応答送受信の例

MACSでは、メッセージを送信する際に必要とする応答の個数を決定しておくので、それ以上の応答は不要である。この不要な応答の送信は、応答送信時のオーバーヘッドとなる。そこで、同報送信を行ったロボットは、必要な数の応答を受信すると finish packet を送信し、他のロボットに応答をする必要がなくなったことを通知する (このパケットによって同報送信に対する応答が終了するものとする)。この finish packet は空スロットを用いて送信するものとし、このパケットが届かなかった場合は (応答が送信され続けるので識別可能) 再び次の空スロットで送信するものとする。

3.4 タイムスロットの割り当て方式

ここでは、タイムスロットの割り当て方式について詳しく述べる。MACS は、同報送信に対する応答が行われている間も、他の同報送信やその同報送信に対する応答の送信をパケットの衝突を起こさずに行うことが可能である。以下にタイムスロットの割り当てプロトコルをまとめておく。

- 最初に同報送信を行う場合は、その同報送信が同期信号になり周囲のロボットが同期する。そして送信メッセージ中の head id で指定されたロボットから順にタイムスロットを与え、応答を送信させる。各々のタイムスロットの後には他のメッセージ送信用の空スロットを割り当てる。
- 空スロットを用いて別の同報送信が行われた場合は、既に応答の送信待ちとなっているロボットが応答の送信を遅らせる。これによって新しい同報送信に対する応答送信スロットを作る。

- 応答の送受信は、メッセージを送信したロボットが、応答を必要な数だけ受信するか、タイムアウトした後に finish packet を送出することにより終了する。
- finish packet を受信した場合は、次の処理をする。
  他に応答受信中のロボットが存在する場合は、次のスロットから finish packet を送信したロボットの応答受信用スロットを取り除く (図4参照)。応答待ちのロボットが存在しなくなった場合は同期が解除される。
- finish packet が他のメッセージのパケットが衝突した場合は、次の空スロットで再送する。

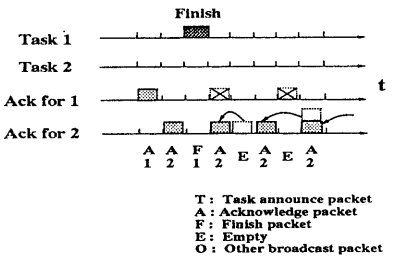

図 4: finish packet が送信された場合 (この場合は、$a_i = 1, a_j = 2$)

なお、空スロットでパケットの衝突が起き、一部のロボットにのみ応答を要する同報送信が伝わった場合は同期にずれが生じる。従って MACS は空スロットで衝突が起きた場合はどのロボットにも受信されないことを前提としている。

タイムスロットの割り当てアルゴリズム

自分のロボット id を $myid$、応答受信中のロボットの台数を rb_{ack}、head id を hid、全体のロボットの台数を $ROBO$、同報送信を受信した時刻を t_g、応答を送信する時刻を t_s、そのメッセージを受信した時刻を t_r、1 スロットにかかる時間を slとする。

まず、応答を送信する時刻 t_sは、

$myid \geq hid$ のとき、

$$t_s = t_g + (myid - hid) \times (rb_{ack} + 1) \times sl$$

$myid < hid$ のとき、

$$t_s = t_g + (ROBO + myid - hid) \times (rb_{ack} + 1) \times sl$$

となる。(+1) は空スロットのスロット長を表している。この式は、$t_s = t_g + wt$ の形で示すことができる。wt は同報送信の受信から応答送信までの待ち時間である。

次にスロットを移動するアルゴリズムについて示す。新たに応答を必要とするメッセージが送信されたとき、応答送信スロット長と、空スロット長が等しく、共に slであるとすると、スロット移動後の新たな送信時間 t'_sは

$$t'_s = t_s + \frac{(t_s - t_r)}{rb_{ack} + 1} \times sl$$

と表される。この式の第 2 項がスロットの移動による遅延を表している。この項の乗算の前部は新たなメッセージを受信した時から応答を送信するまでのスロットサイクル数を表している。すなわち第 2 項は新たなメッセージを受信した時刻での自分が応答を送信するまでのスロットサイクル数を計算し、その値に 1 スロットの長さを乗算した値である。また、後から送信されたメッセージに対して応答を送信するロボットの送信時刻 t_{snew}は

$$t_{snew} = t_s + (rb_{ack} - 1) \times sl$$

となる (ただし、応答を受信するロボットが 1 台増えるので rb_{ack}の値は同報送信受信前に比べて 1 増えている)。

次に、finish packet の送信によるタイムスロットの圧縮のアルゴリズムを示す。圧縮の時間を算出するためには、その時点で応答受信中のロボット数と、それらのロボットがメッセージを送信した時刻に基づく、スロットサイクル内での応答の送信順序が分かれば良い。

finish packet を送信したロボットのスロットサイクル内での応答送信の順番を a_i、スロットを圧縮するロボットのサイクル内での順番を a_jとすると (図 4の例を参照)、finish packet 受信にともなう新たな送信時刻 t'_sは、

$a_i > a_j$のとき

$$t'_s = t_s - \frac{(t_s - t_r)}{rb_{ack} + 1} \times sl$$

$a_i < a_j$のとき

$$t'_s = t_s - (\frac{(t_s - t_r)}{rb_{ack} + 1} + 1) \times sl$$

となる。

4 実装

この章では、MACS の実装方法について述べる。本研究では、我が研究室で開発した自律移動ロボット Einstein を利用し実装を行った。Einstein の外観を図

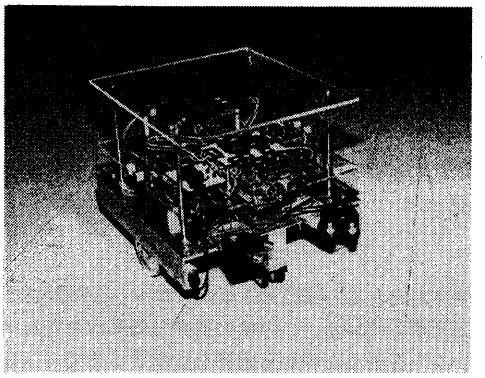

図 5: Einstein の外観

5に示す。Einstein は2つの駆動輪を持ち、パルスモータで駆動する。また、外部との通信手段として無線による通信機構 (無線通信機＋ TNC(1200bps)) を持つ。更に外界の状況を把握する手段として4方向に超音波センサを、そして物の受渡しをする手段として電磁石アームを持つ。そして、これらのデバイスを制御するための CPU ボードが搭載されており、我が研究室で開発したマルチタスク・マルチスレッド OS PULSER [4] が実装されている。PULSER は、ロボットのリアクティブな行動を取り扱うために開発された OS であり、センサ値の変化や無線を介した通信などのイベントに対し、リアクティブに行動することができる。

MACS は C 言語を用いて実装されており、ユーザにはタスクの宣言をする関数が提供される。ユーザがアプリケーションを書く際には、その関数をプログラム中から呼び出す形で利用する。図 6にアプリケーションの例を示す。この例は、タスクの宣言を行った後で返ってきた応答メッセージの内容を表示する関数である。init() は、応答リストやその他の変数を初期化する関数である。ユーザがタスクの宣言を行うときは、task_announce という関数を用いる。task_announce には、4 つの引数がある。それらは順に、タスク id、タイムアウトするまでの時間 (sec)、必要とする応答の数、メッセージの内容である。メッセージの内容は文字列に限定している。この例の場合は、「id は 15、内容は "task" で、必要とする応答の数が 3 という条件で 10 秒間応答を受信する」という場合を表している。

この関数を実行すると、内部で現在の応答受信待ちリストを見て head id を決定し、同報送信を行う。パケットの送信は 1 度では受信されないことがあるので 3 回送信する。そして、受信側が最初に受信したパケットがその受信した側にとっての同期信号となる。そして、応答が必要な数だけ返送されるかタイムアウトした後、返送された応答のリストを図 7に示す b_mail 構造体の型で返す。応答の内容を見る時は、status というメンバを参照する。メッセージ部の仕様は、アプリケーションの種類に応じてユーザが自由に決定できる。

```
void example(){
    int i;
    b_mail *ret;

    init();          /* 変数の初期化 */

    ret = task_announce(15,10,3,"task");

    while( ret->mstype != NULL ){
        printf( "%s -> ", ret->status );
        ret = ret->next;
    }
    printf( "NULL\n" );
}
```

図 6: アプリケーションの例

```
typedef struct _bmail{
    int  id;       /* 送信者の id */
    char mstype;   /* メッセージの種類 */
    char acktype;  /* 応答の必要性 */
    int  tid;      /* タスク id */
    char *status;
         /* メッセージの内容 */
    struct  _bmail    *next;
         /* リストの次要素へのポインタ */
}b_mail;
```

図 7: b_mail 構造体

5 評価

この章では、実際に実装した MACS を用いて、MACS プロトコルの妥当性、及び性能を評価する。

MACS の応答機構では、各々のロボットが独立に同報送信を受信した時刻から、自分の待ち時間だけ待機した後に応答を送信する方法をとっている。また、応答以外の送信を行うときも、空スロットの順番がくる時刻を最初の同報送信を受信した時刻から計算して送信している。ここで注意すべきことは、各々のロボットが独立にタイムスロットを刻んでいるので互いにタイムスロットがずれている可能性があるという点である。従って各ロボットの同期がとれているかどう

かを確認する必要がある。同期がずれると図8に示すようなパケットの衝突が起こる可能性がある。

また、タイムスロットの幅が短すぎてもパケットの衝突が起こる。しかし、逆にタイムスロットの幅が長すぎると通信媒体の利用効率が下がってしまう。

そこで、実際にMACSを用いた実験を行いタイムスロットのずれを評価し最適なタイムスロット幅を検討する。なお、評価に用いたTNC（パケット通信用のモデム）は1200bpsで、Einsteinが搭載しているOS PULSERは10msのタイムスライスでパケットの着信をチェックしている。また応答のパケット長は最大50バイトとする。

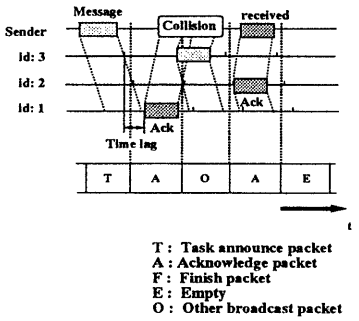

図 8: タイムスロットのずれの影響

最初に、送信パケット長に対する、同報送信を行なってから最初の応答を受信するまでの時間（ターンアラウンド時間）の関係を図9に示す。なお、同報送信のパケット長は30バイトであり、応答は10個送信した。

図9から、ターンアラウンド時間は応答パケット長にほぼ比例して大きくなっている。同報送信のパケット長が30バイトであるので、同報送信をする側がターンアラウンド時間の中に占める時間は、応答パケット長が30バイトの時のターンアラウンド時間が4.5(sec)であるので、ほぼ $4.5/2 = 2.25$(sec) であると考えられる。なお、PULSER上のプログラムに余計なスレッドを5つ走らせ、負荷を高めてターンアラウンド時間を計ってみたところ、ターンアラウンド時間はほとんど変わらなかった。通常PULSER上で実行するプログラムのスレッドは3個くらいであるので、負荷の大小による影響はほとんどないものと思われる。

次に応答パケット長を20バイトにした時の、タイムスロットの幅に対するパケットの到着率（パケットがタイムスロット通りに到着する確率）を図10に示す。

図10を見ると、タイムスロットの幅が1.5(sec)の時はパケットの衝突が起きているが2.0(sec)以上の時にパケットが衝突を起こすことなく送信されていることがわかる。従って、この図から応答パケット長が20バイトの時の最適なタイムスロット幅は1.5～2.0(sec)の間の値であることがわかる。

これは、ターンアラウンド時間の中に同報送信が占める時間がほぼ2.25(sec)であることを応答パケット長が20バイトの時のターンアラウンド時間(4.0 sec)に当てはめてみた結果 $(4.0 - 2.25 = 1.75$ sec) とほぼ一致する。また、同様に応答パケット長が30バイト以上のときについても最適なタイムスロット幅を求めた。その結果、同様に応答パケット長が30,40バイトの時の最適なスロット幅は2.0～2.5(sec)の間で、応答パケット長が50バイトの時は2.5～3.0(sec)であった。

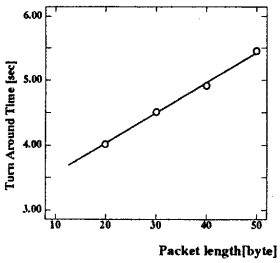

図 9: 応答パケット長に対するターンアラウンド時間

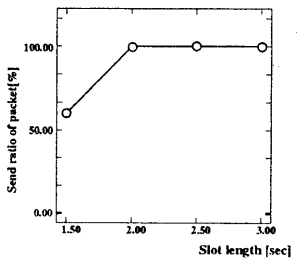

図 10: タイムスロットの幅に対するパケットが衝突せずに到着する確率（メッセージ長20バイトのとき）

6 おわりに

本研究では、自律移動ロボット間で Contract Net を用いたアプリケーションを作成する際に必要となる、同報送信に対する応答機構 MACS を設計し、我が研究室で開発したロボット用 OS PULSER 上に実装した。

MACS は、応答送信をするロボットに対し、タイムスロットを割り当てて応答を送信させることにより、衝突を起こすことなく応答送信を行うことができる。さらに、空スロットを導入しスロットの移動を可能にすることにより、同報送信に対する応答が行われている間も、他の同報送信に対する応答を行うことが可能である。この機構により、同時に複数の Contract Net を起動することが可能である。

また、実験の結果、同報送信を受信するロボット間でのスロットのずれはほとんどなく、同期がとれなくなる心配はほとんどないことが分かった。

今後の課題としては、空スロットに複数のメッセージが送信されることによるパケットの衝突や隠れ端末への対処をすること、マルチキャストの宛先指定方法の改善、及びマルチホップへの拡張が挙げられる。

参考文献

[1] 石田慶樹, 浅間一, 松元明弘, 尾崎功一, 遠藤勲 :自律分散型ロボットシステム ACTRESS（第 3 報）– 通信に基づく協調的問題解決の戦略, 第 8 回ロボット学会学術講演会予稿集, 1990, pp.883-884.

[2] 久保田浩司, 木下研作 : バス型 LAN における同報応答方式に関する研究, 信学技報 RCS88-27

[3] 斎藤利忠, 所真理雄 : 同報送信に対する Ack 機能を備えた CSMA/CD 通信方式 GAck Ethernet の提案, 電子通信学会論文誌 Vol.1 J71-B No.4, 1988,pp523-532.

[4] 菅原 智義, 矢向 高弘, 安西 祐一郎 : 自律移動ロボット用 OS PULSER の設計と実装, '92 情報処理学会 第 44 回（平成 4 年前期）全国大会 講演論文集, 1992.

[5] I. Cidon, M.Sidi : Distributed assignment algorithms for multi hop packet-radio networks, *IEEE INFOCOM*, 1988, pp.1110-1117.

[6] A. Ephremides, T. Truong : Distributed algorithm for efficient and interference-free broadcasting in radio networks, *IEEE INFOCOM '88*, 1988, pp.1119-1124.

[7] R.Ramaswami, K.K.Parhi : Distributed scheduling of broadcast in a radio network, *IEEE INFOCOM '89*, 1989, pp.497-504.

[8] L.C.Pond, V.O.K.Li : Adaptive signalling in distributed self-organizing mobile packet radio networks, *IEEE Internal Conference on Communication*, 1990, pp331.1.1-331.1.5.

[9] R.Smith : The contract net protocol: High level communication and control in a distributed problem solver, *IEEE Transaction on Conputers* ,C-29(12):1104-1113.December 1980.