行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※　具地域性與行動性之分散式資源分配問題探討　※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☑個別型計畫　　□整合型計畫
計畫編號：NSC　90－2213－E－009－150－
執行期間：　90 年　8 月　1 日至　91 年　7 月　31 日

計畫主持人：黃廷祿
計畫參與人員：陳勝雄

本成果報告包括以下應繳交之附件：
- □赴國外出差或研習心得報告一份
- □赴大陸地區出差或研習心得報告一份
- □出席國際學術會議心得報告及發表之論文各一份
- □國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊工程學系

中　華　民　國　　91 年　10　月　28　日

行政院國家科學委員會專題研究計畫成果報告
具地域性與行動性之分散式資源分配問題探討
Resource Allocation in Distributed Systems with Locality and Mobility

計畫編號：NSC 90-2213-E-009-150
執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：黃廷祿　國立交通大學資訊工程學系
E-mail: tlhuang@csie.nctu.edu.tw
計畫參與人員：陳勝雄　國立交通大學資訊工程學系
E-mail: chenss@csie.nctu.edu.tw

一、中文摘要

資源分配問題是共享記憶體系統及分散式系統中一個很基本的問題。隨著無線網路技術的進步，使用者可以藉由無線方式使用資源。為了解決在行動隨意網路中的資源分配問題，本計畫提出一個更根本的演算法—針對行動隨意網路的wave演算法。藉由此wave演算法可以架構出互斥問題的演算法。利用互斥問題演算法即可用來解決資源分配的問題。此wave演算法的正確性並不需要假設整個網路架構是永遠連接（always connected）。只需要網路架構符合一致性連接（consistently connected），亦即只要保證每個點（process）送出的訊息最終都會被所有其他的點所收到。

關鍵詞：資源分配問題、Wave 演算法、行動計算、分散式演算法

Abstract

Resource allocation problem is a fundamental problem in shared memory systems and distributed systems. With the advances in wireless networking technology, resource sharing via air became available. In order to resolve this problem in mobile networks, a more basic problem, called wave algorithms, has been studied in this project. Based on a wave algorithm, we are able to construct a mutual exclusion algorithm which can be used to resolve conflicting accesses to shared resources. This project presents a wave algorithm for mobile ad hoc networks in which links may fail or reform. Without assuming that the network topology is always connected, the algorithm only requires the network to be *consistently* connected, a very mild condition imposed on the network ensuring that each message flooded by a process will be received by all processes eventually.

Keywords: Resource Allocation, Wave Algorithm, Mobile Computing, Distributed Algorithm

二、緣由與目的

Resource allocation problem is a fundamental problem in distributed systems. With the advances in wireless networking technology, resource sharing via air became available. This project aims to provide an algorithm to resolve the resource allocation problem in mobile networks.

A mobile network is defined as a collection of mobile platforms or nodes where each node is free to move about arbitrarily [3]. A pair of nodes communicates by sending messages either over a direct wireless link, or over a sequence of wireless links including one or more intermediate nodes. A pair of nodes can communicate directly only if they lie within one another’s transmission radius. A link forms between a pair of nodes when nodes move into one another’s transmission radius; in contrast, a link fails when nodes move out of one another's transmission radius.

Due to link failures and link formations, designing distributed algorithms for mobile networks is a challenging task. In order to resolve resource allocation problem, this project studies a more fundamental problem, called wave algorithms, for mobile networks. A wave algorithm ensures the participation of all processes and has been known as a useful building block in distributed systems. For example, it can be used for some fundamental tasks, e.g. broadcasting [4], etc. In addition, wave algorithms can be used in more complicated problems such as leader election, termination detection, and mutual exclusion [5]. A mutual exclusion algorithm can be used to resolve conflicting accesses to shared resources.

In every wave process, there is a special type of internal event called a *decide* event. A wave algorithm exchanges messages and then the algorithm makes at least one decision, which depends causally on some event in each process. A process is an *initiator* if it starts the execution of its local algorithm spontaneously; in contrast, a *non-initiator* becomes involved in the algorithm only when a message of the algorithm arrives and triggers the execution of the process algorithm.

A wave algorithm is called *centralized* (e.g., [1,4]) if there must be exactly one initiator in each execution, and *decentralized* (e.g., [2,6]) if the algorithm can be started spontaneously by an arbitrary subset of the processes. A decentralized algorithm is more general. However, more messages are needed.

Previous wave algorithms work correctly only if the network is always connected and each link is permanent. In this project, we present a decentralized wave algorithm tolerating link failures and link formations.

三、結果與討論

We consider a distributed system consisting of a finite set P of independent mobile nodes, communicating by message passing over a wireless network. A link between two nodes indicates they are within one another's transmission radius. Assumptions on the mobile nodes and network are:

1. the nodes have unique node identities,
2. node failures do not occur,
3. communication links are bidirectional,
4. neighbor-awareness, that is, a link-level protocol ensures that each node is aware of the set of nodes with which it can currently directly communicate by providing indications of link formations and failures,
5. the network is connected initially,
6. the network is *consistently* connected.

Next, we assume that each node has a wave process, modelled as a state machine, with a set of states, some of which are initial states, and a transition function. The transitions are associated with named *events*. The events are classified as either *internal*, *input*, or *output*. The inputs and outputs are used for communication with the environment, while the internal actions are visible only to the process itself. The internal events at node p_i include the ones below.

- Init_{p_i}: if node p_i is an initiator, this event is enabled spontaneously to start the wave process.
- Decide_{p_i}: the decide event of node p_i.

Input events are as follows.

- $\text{Recv}_{p_i}(p_j,m)$: node p_i receives message m from node p_j.
- $\text{LinkUp}_{p_i}(p_j)$: node p_i receives notification that the link between p_i and p_j is now up.
- $\text{LinkDown}_{p_i}(p_j)$: node p_i receives notification that the link between p_i and p_j is now down.

The transition function takes as input the current state of the process and the input or internal event, and produces as a (possibly empty) set of output events and a new state for the process. Output event is:

- Send$_{p_i}$(p_j,m): node p_i sends message m to node p_j.

A wave algorithm is an algorithm that satisfies the following conditions.

1. **Decision**. Each execution contains at least one decide event.
2. **Dependence**. In each execution each decide event is causally preceded by an application event in each other process.

Wave algorithm for MANETs

Finn's algorithm [2] is a wave algorithm that can be used in arbitrary networks but doesn't tolerate link failures and link reformations. We adapt Finn's algorithm so that it works in mobile ad hoc networks (MANETs).

In Finn's algorithm, each process p maintains two sets of process identities, Inc_p and $NInc_p$. A process q is in Inc_p if an event in q precedes the most recent event in p, and in $NInc_p$ if for all neighbors r of q an event in r precedes the most recent event in p. The basic action of each process p is sending messages, including Inc_p and $NInc_p$, to neighbors whenever one of this two sets has increased. Initially $Inc_p = \{p\}$ and $NInc_p = Ø$. When p receives a message, containing Inc and $NInc$ sets, the received identities are inserted into p's versions of these sets. After receiving a message from all neighbors (i.e., $Neigh_p \subseteq Inc_p$), p is inserted into $NInc_p$. Decide$_p$ doesn't enabled until $Inc_p = NInc_p$. Since the network is connected, if the condition $Inc_p = NInc_p$ holds, Inc_p has contained all processes in the network (which can easily be proved by induction), and then Decide$_p$ can be enabled.

However, the network may be partitioned because each node has mobility in MANETs. In this case, it is possible that a process at some partition decides before its decide event is causally preceded by an event of each process. In order to solve this problem, each process maintains an additional set of process identities, $Leave_p$, consisting of process q such that no event in q precedes the most recent event in p and q is no longer p's neighbor. Process decides only if $Inc_p = NInc_p$ and $Leave_p = Ø$. Thus, the problem is avoided.

The wave algorithm is event-driven. Actions triggered by an event are assumed to be executed atomically. The pseudocode triggered by Init$_p$ and Recv$_p$ is shown in Fig. 1. The pseudocode triggered by LinkDown$_p$ and LinkUp$_p$ is shown in Fig. 2.

四、計畫成果自評

為了解決無線環境中資源分配問題，本計畫提出了更根本的 wave 演算法。基於此演算法，可以架構出 leader election、mutual exclusion 等問題的演算法。而 mutual exclusion algorithm 即可用來處理資源分配問題。

在無線環境中幾乎所有傳統的分散式系統的問題都無解。因此，如何界定問題有解與無解的界線以及如何對系統作假設就顯得極具學術價值。本計畫成果在嚴謹地界定上稍弱，加強之後即可在學術期刊中發表。

五、參考文獻

[1] E.J.-H. Chang. Echo Algorithms: Depth Parallel Operations on General Graphs. *IEEE Trans. Sofew. Eng.*, vol. SE-8, no. 4, pp. 391-401, July 1982.

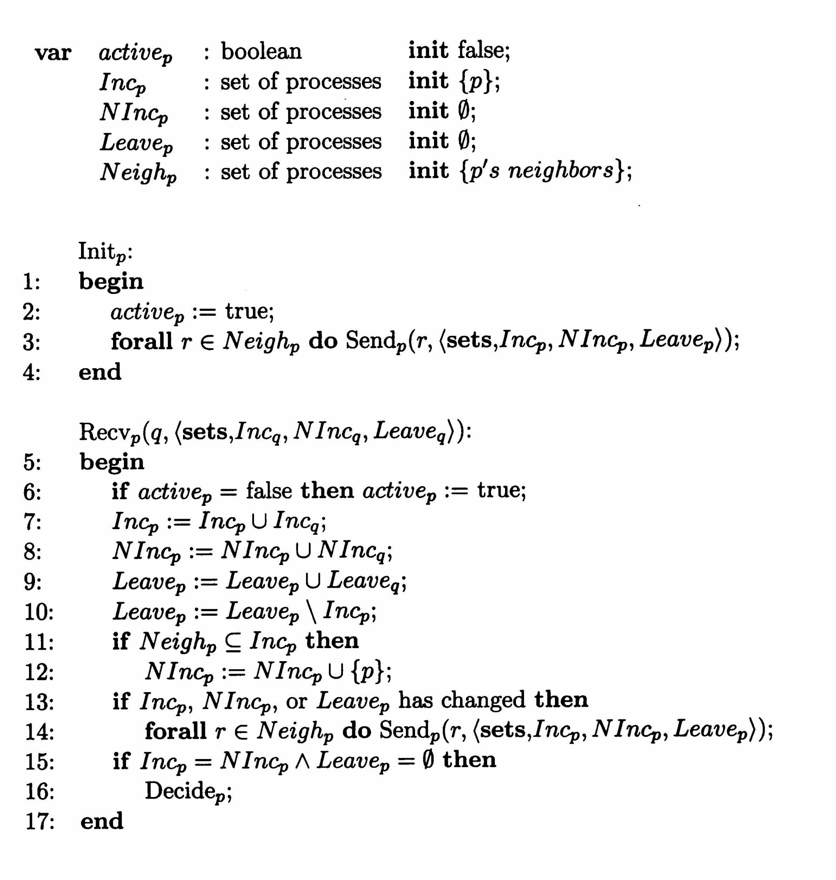
[2] S.G. Finn. Resynch Procedures and Fail-safe Network Protocol. *IEEE Trans. Commun.*, vol. COM-27, no.6, pp.840-845, June 1979.

[3] J. Macker and M.S. Corson. Mobile Ad Hoc Networking and the IFTF. *ACM Mobile Computing and Communication Review* 2(1), pp.9-14, Jan. 1998.

[4] A. Segall. Distributed network protocols. *IEEE Trans. Inf. Theory*, vol. IT-29, pp.23-35, 1983.

[5] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2000.

[6] G. Tel. *Topics in Distributed Algorithms*. vol. 1 of Cambridge Int. Series on Parallel Computation, Cambridge University Press, 1991.

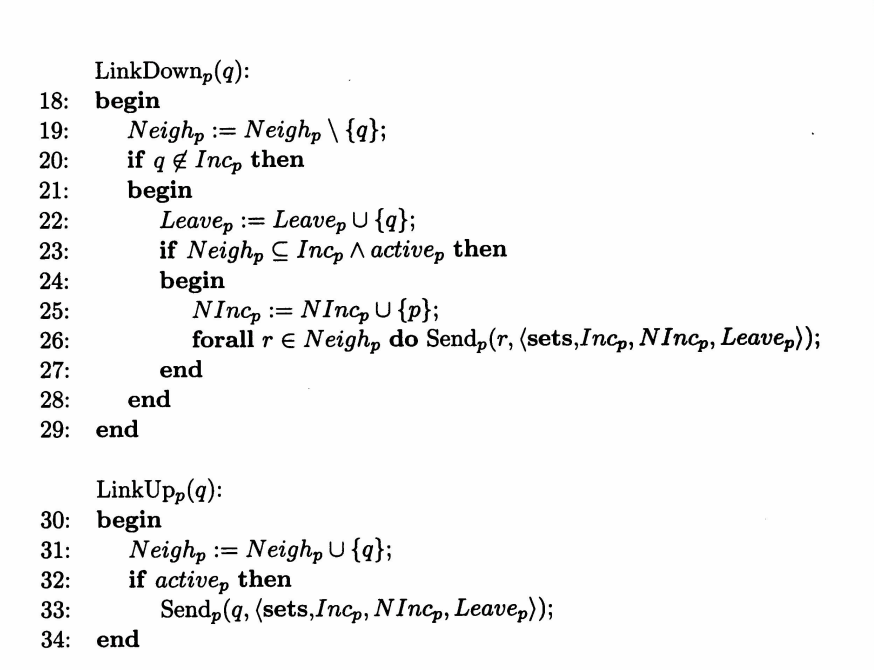


```
var $active_p$   : boolean           init false;
    $Inc_p$      : set of processes  init $\{p\}$;
    $NInc_p$     : set of processes  init $\emptyset$;
    $Leave_p$    : set of processes  init $\emptyset$;
    $Neigh_p$    : set of processes  init $\{p's\ neighbors\}$;

    Init$_p$:
    begin
        $active_p$ := true;
        forall $r \in Neigh_p$ do Send$_p(r, \langle$sets$, Inc_p, NInc_p, Leave_p\rangle)$;
    end

    Recv$_p(q, \langle$sets$, Inc_q, NInc_q, Leave_q\rangle)$:
    begin
        if $active_p$ = false then $active_p$ := true;
        $Inc_p := Inc_p \cup Inc_q$;
        $NInc_p := NInc_p \cup NInc_q$;
        $Leave_p := Leave_p \cup Leave_q$;
        $Leave_p := Leave_p \setminus Inc_p$;
        if $Neigh_p \subseteq Inc_p$ then
            $NInc_p := NInc_p \cup \{p\}$;
        if $Inc_p$, $NInc_p$, or $Leave_p$ has changed then
            forall $r \in Neigh_p$ do Send$_p(r, \langle$sets$, Inc_p, NInc_p, Leave_p\rangle)$;
        if $Inc_p = NInc_p \wedge Leave_p = \emptyset$ then
            Decide$_p$;
    end
```

Figure 1: Pseudocode triggered by Init$_p$ and Recv$_p$ events.



```
    LinkDown$_p(q)$:
    begin
        $Neigh_p := Neigh_p \setminus \{q\}$;
        if $q \notin Inc_p$ then
        begin
            $Leave_p := Leave_p \cup \{q\}$;
            if $Neigh_p \subseteq Inc_p \wedge active_p$ then
            begin
                $NInc_p := NInc_p \cup \{p\}$;
                forall $r \in Neigh_p$ do Send$_p(r, \langle$sets$, Inc_p, NInc_p, Leave_p\rangle)$;
            end
        end
    end

    LinkUp$_p(q)$:
    begin
        $Neigh_p := Neigh_p \cup \{q\}$;
        if $active_p$ then
            Send$_p(q, \langle$sets$, Inc_p, NInc_p, Leave_p\rangle)$;
    end
```

Figure 2: Pseudocode triggered by LinkDown$_p$ and LinkUp$_p$ events.