Online Ring Structure Adaptation for Mobile Wireless Computing

Li-Hsing Yen
Dept. Computer Science & Information Engineering
Chung Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: lhyen@chu.edu.tw

Kuang-Hui Chi
Dept. Electrical Engineering
National Yunlin University of Science & Technology
Touliu, Taiwan 640, R.O.C.
Email: chikh@yuntech.edu.tw

***Abstract*— In a mobile *ad hoc* (multi-hop) wireless network, the logical structure of a ring is likely to become volatile or expensive to maintain over time due to changeable network topology. Additional adverse effects take place when a node joins or leaves the computation in the presence of mobility. This paper presents a protocol that adapts a ring among mobile nodes to the network dynamics to reflect overall communication efficiency. This is achieved by modifying the ring structure in a localized, mutual exclusive fashion, thereby allowing for concurrent segment-wise modifications to proceed. Remarkably our proposal operates without global knowledge of the logical structure and can be embodied as an underlying protocol stratum that supports transparent deployments of conventional algorithms in mobile environment. Subsequent to correctness proof, simulation results show that our proposal is promising in several regards.**

I. Introduction

Due to the development of wireless technology and portable computing devices, mobile *ad hoc* (multihop) wireless networks have drawn considerable attention from the research community. This type of network, unlike conventional infrastructured wireless counterparts, operates in the absence of fixed switching stations and thus all networking entities therein can be mobile. In order to maintain system-wide communication, each host may serve as an intermediate node to relay messages. Applications such as disaster recovery, crowd control, search and rescue, and automated battlefields are typical examples of where ad hoc networks are deployed.

To date, study in this field mainly focuses on medium access control [1] and routing [2], [3]. Instead, we are concerned with maintaining a ring structure among mobile nodes that adapts to changeable network topology. Rings are widely used for distributed control computations like leader election [4], [5], [6], [7], multicast [8], [9], mutual exclusion [10], or termination detection because of simple connectivities. In this text, a ring refers to an embedded logical structure spanning independently from the physical network topology and is not necessarily a subgraph thereof. A ring may remain unchanged even though the network has altered physical connectivities due to participant host migrations.

Fig. 1 shows two possible rings overlaid to an ad hoc network. It can be seen that the correctness of a ring structure is defacto orthogonal to the underlying network topology. Nevertheless, a structure in line with physical linkage favors

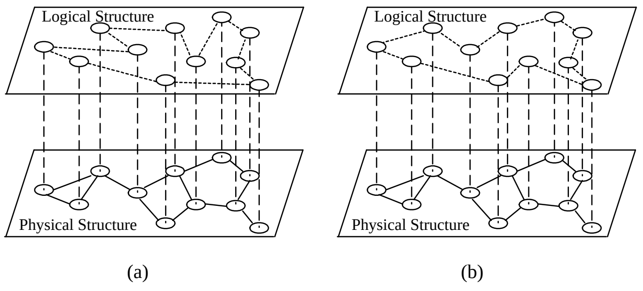


Fig. 1. Two rings on the same physical network topology.

communication as well as computation to an extent, e.g., reducing message-circulation time of a Token Ring. This paper addresses the performance aspects of logical structure in an ad hoc network. In the context of infrastructured mobile networks, Badrinath *et al.* [11] identified similar issues and suggested that a logical structure be confined on the fixed part of the network. Following the thread, researchers adapted ring- or tree-based distributed algorithms for mobile nodes [12], [13]. Their derived results, however, do not apply to ad hoc settings in the absence of stationary devices.

Given the communication cost of each link, finding a minimum-cost ring is essentially the traveling salesperson problem and thus NP-hard. Further, benefits from an optimal structure are likely overwhelmed by repeatedly invoking an algorithm that involves all nodes even merely on a slight change of network status. This may lead to substantial system resource (power, bandwidth, etc) consumption, making an optimal solution untenable in highly mobile environment. Moreover, an optimal algorithm may not terminate if restarted whenever the network changes partly.

We do not aim at maintaining a globally but a locally optimal ring structure in response to host movements or participant changes. In this light, we propose to modify the logical structure only when the network has changed some certain degree. Each such modification is restricted within a single region of four nodes, without the awareness of the entire system. Our protocol fabrics are illustrated in Fig. 2 where solid lines represent directed edges of the ring, each labeled with message-delivery cost. We observe that interchanging

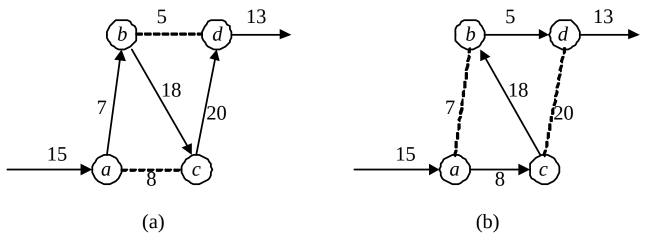


Fig. 2. Illuminating the basic idea behind our approach. (a) Original arrangement. (b) After a local modification.

logical positions of nodes b and c in Fig. 2(a) benefits overall communication (see Fig. 2(b) for a result.) Since our scheme involves a small number of nodes, resource and execution time can be thus saved significantly.

Our proposal enables prescribed segment-wise modifications to proceed concurrently. Nonetheless, concurrent updates to overlapping segments require arbitration. For this, we further develop an algorithm whereby nodes acquire a localized, mutual exclusive authority for modification activities. The algorithm is shown both safe and deadlock-free. Our overall development is intended for a communication substratum underlying conventional protocols so that protocols designed for stationary hosts remain viable in mobile ad hoc environment.

The rest of this paper is organized as follows. In the next section, we will present the ring modification procedure and the localized mutual-exclusion protocol that deals with concurrent modifications. Section III proves the correctness of the proposed mutual-exclusion protocol. Performance evaluation results are given in Section IV. Lastly, Section V draws our concluding remarks.

II. The Protocol

To set the stage for subsequent development, we make assumptions as follows. Each node N has a unique identity and is aware of its preceding, immediate succeeding, and second succeeding nodes (termed *N.pred*, *N.succ*, and *N.succ2*, respectively.) There are hosts serving as intermediate routing nodes such that N can be mutually reached from *N.pred*, *N.succ*, and *N.succ2*. Communication costs with the three neighboring nodes are measured as well. To simplify the discussion, we assume that message delivery between N and any of these neighbors is reliable and in order, which can be achieved by, e.g., TCP. We impose neither restrictions on the speed and direction of host movements, nor on the number of participating processes. Each process is allowed to join or leave a computation at its will.

A. Base Procedure

In intuitive terms, re-arranging a ring of nodes is suggested if the performance gain outweighs its execution cost. Performance gain refers to the difference of cost for transferring a message over the ring before and after topological changes. This measure can be derived next, considering that communication cost between two nodes may differ bidirectionally.

Let *Cost*(M, N) represent the communication cost from nodes M to N. Any node N sends detected *Cost*$(N$,*N.pred*$)$ and *Cost*$(N$,*N.succ*$)$ to its *N.pred*, and collected *Cost*$(N$,*N.succ*$)$ and *Cost*$(N$,*N.succ2*$)$ to *N.succ*. Such message exchanges enable N to learn *Cost*$($*N.succ*$, N)$, *Cost*$($*N.succ*,*N.succ2*$)$, *Cost*$($*N.pred*$, N)$, and *Cost*$($*N.pred*,*N.succ*$)$ from neighboring sites. Node N thereby determines whether a switch from topologies $\{$*N.pred*, *N.succ*, N, *N.succ2*$\}$ to $\{$*N.pred*, N, *N.succ*, *N.succ2*$\}$ is cost-effective. That is, the former demands

$$\mathcal{C} = \textit{Cost}(\textit{N.pred}, N)+\textit{Cost}(N, \textit{N.succ})+\textit{Cost}(N.succ, N.succ2)$$

whereas the latter

$$\mathcal{C}' = \textit{Cost}(\textit{N.pred}, \textit{N.succ})+\textit{Cost}(\textit{N.succ}, N)+\textit{Cost}(N, \textit{N.succ2})$$

If $\mathcal{C} > \mathcal{C}' + \delta$, where $\delta > 0$ represents the cost of executing the proposed protocol, N initiates connectivity modifications in the ring by swapping its position with *N.succ*. Indeed, we can subsume in δ an additional threshold by which $\mathcal{C}$ outgrows $\mathcal{C}'$, indicating the minimum performance gain of actual interest to the system.

We introduce a message *Set*$($*Pred*,*Succ*,*Succ2*$)$ to update the recipient knowledge of its preceding, succeeding, and second succeeding nodes. A parameter absent from actual use by the message is denoted as an underscore ('_') for short. To initiate structural modifications, node N executes steps below.

1) Send *Set*$($*N.pred*,N,*N.succ2*$)$ to *N.succ*.
2) Send *Set*$($_,*N.succ*,$N)$ to *N.pred*.
3) Send *Set*$(N, \_, \_)$ to $N.succ2$.
4) *N.pred* $\leftarrow$ *N.succ*
5) *N.succ* $\leftarrow$ *N.succ2*

On receipt of *Set* message, a node, including N itself, executes the protocol of Fig. 3.

```
On receiving Set($W, X, Y$) from node $Z$
        if ($W \neq$ '_') then
                M.pred $\leftarrow W$
        if ($X \neq$ '_') then
                M.succ $\leftarrow X$
        if ($Y \neq$ '_') then
                M.succ2 $\leftarrow Y$
        if ($W =$ '_' and $X \neq$ '_') then
                Send Set(_, _, $M.succ$) to M.pred
        if ($Z = W$) then
                Send Set(_, _, $M.succ$) to $Z$
```

Fig. 3. Protocol for structural modifications

An application of our protocol over nodes $\{P, Q, R, S, T, U\}$ is depicted in Fig. 4. Table I summarizes how corresponding nodes alter connectivities before and after executing the protocol (first two columns.)

B. Tackling Concurrent Modifications

We allow concurrent modifications of a ring, which, however, cannot be conducted arbitrarily. Otherwise, incorrect

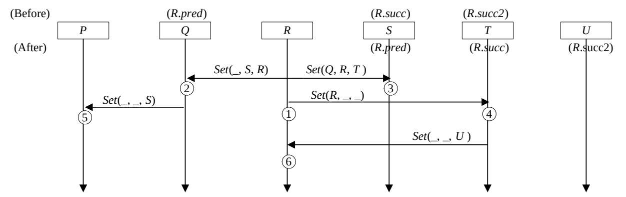


Fig. 4. A scenario of applying our protocol. R initiates a topological change.

TABLE I
CHANGE OF CONNECTIVITIES IN FIG. 4. THE LAST COLUMN INDICATES THE TIME INSTANTS OF THESE EVENTS NUMBERED IN THE FIGURE.

Before	After	Time instant
P.succ2 $= R$	*P.succ2* $= S$	5
Q.succ $= R$	*Q.succ* $= S$	2
Q.succ2 $= S$	*Q.succ2* $= R$	2
R.pred $= Q$	*R.pred* $= S$	1
R.succ $= S$	*R.succ* $= T$	1
R.succ2 $= T$	*R.succ2* $= U$	6
S.pred $= R$	*S.pred* $= Q$	3
S.succ $= T$	*S.succ* $= R$	3
S.succ2 $= U$	*S.succ2* $= T$	3
T.pred $= S$	*T.pred* $= R$	4

structure may occur. For instance, nodes b and c of Fig. 5(a) activate modifications concurrently, resulting in incorrect logical structures. Let us take the result of Fig. 5(c) as an explanation. Note first only the *pred* pointers of b, c, and d will be changed by b's modification activation and only those of c, d, and e will be changed by c's. When b and c activate modifications concurrently, *b.pred* will be set to c as a part of b's protocol initiation steps while *e.pred* will be set to c due to the reception of c's *Set* message. Both b's and c's *Set* messages instruct d to change its *pred* pointer to b, so the result is deterministic despite of race condition. Finally, *c.pred* will first point to d as a part of c's protocol initiation steps and later be changed to a due to the reception of b's *Set* message. The final result is therefore obtained.

As a remedy for the problem, we further contrive a mechanism that mediates structural updates in a localized, mutual exclusive fashion. We essentially preclude two nodes N and M in close vicinity such that $\{$*N.pred*, *N.succ*, *N.succ2*$\} \cap \{$*M.pred*, *M.succ*, *M.succ2*$\} \neq \emptyset$ from initiating concurrent topological adjustments. One may question if this restriction is too strong. In particular, one may suspect that P and S in any ring segment $\{P, Q, R, S, T\}$ can safely initiate the protocol without any harm, though $R =$ *P.succ2* $=$ *S.pred*. To explain, note that a topological change will modify the initiating node's *succ2* pointer as well as the *succ2* pointers of the initiating node's successor, predecessor, and second predecessor. Therefore, when P and S initiate the protocol concurrently, *Q.succ2* can be changed to R due to P's action and can also be changed to T due to S's, a harmful race condition.

This problem reduces to distributed dining philosophers problem [14] (diners problem), which demands mutual exclusion only among *neighboring* nodes, in the following way.

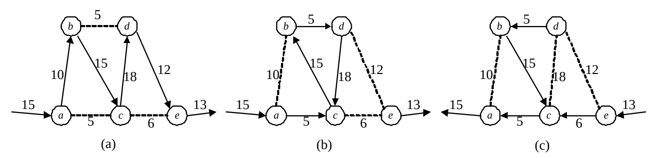


Fig. 5. Illustrating concurrent structural modifications. (a) Original arrangement. (b) Incorrect result (arrowed lines indicate *succ* pointers). (c) Incorrect result (arrowed lines indicate *pred* pointers).

Given a ring of n nodes, we can construct a undirected graph $G = (V, E)$ where V represents the set of all nodes in the ring and edge (u, v) is in E if and only if nodes u and v cannot perform adjustments concurrently. This forms a Harary graph $H_{6,n}$ [15], the smallest 6-connected graph with n vertices. In this way, the problem at hand reduces to the diners problem on a Harary graph. Solutions to the diners problem on a Harary graph, however, are not equally viable here for the following reasons.

- Observed solutions to the diners problem require message exchange facility between neighboring nodes. In contrast, a node in our Harary graph is not aware of all its neighbors, so such facility does not necessarily exist.
- Our architecture does not require every request to be eventually satisfied as with the diners problem. We disable some of the conflicting requests; nodes with unfulfilled requests depart from exclusion contention and, after re-assessment, can issue requests subsequent to current topological changes.

We thus develop a new algorithm suited to our setting. To resolve contending nodes, we take a similar strategy that many distributed mutual exclusion algorithms exploit: define a total ordering relation on a pair of node's identity and some other value. Each node is associated with a tuple $\langle m, n \rangle$, where m quantifies the benefits to be gained after triggering a modification and n represents the node's unique identity. For any $\langle m, n \rangle$ and $\langle m', n' \rangle$, we have $\langle m, n \rangle \prec \langle m', n' \rangle$ if and only if

$$m < m' \text{ or } (m = m' \text{ and } n < n')$$

In this case, the node with $\langle m', n' \rangle$ has higher priority.

At any time each node operates in one of six states: *WHITE*, *GRAY*, *RED*, *BLUE*, *GREEN*, or *BLACK*. Initially *WHITE*, the current state of node N is recorded in a local variable, denoted by *N.color*. Only when *N.color* is *WHITE*, N is qualified to request the authority for initiating structural modifications:

1) Quantify the benefits to be gained from the modification into *N.gain*.
2) Send message *RedReq* to *N.pred*, *BlueReq* to *N.succ*, and *GreenReq* to *N.succ2*, all messages tagged with $\langle$*N.gain*, *N.id*$\rangle$.
3) *N.color* $\leftarrow$ *GRAY*.

A recipient node M in the *WHITE* state accepts the request through the *acceptance* procedure:

1) Change *M.color* to the received message's type (*RED*, *BLUE*, or *GREEN*) and record the tagged $\langle$*N.gain*, *N.id*$\rangle$.

2) Reply N with *ReqAcpt*.

After having received three *ReqAcpt* messages, node N in the *GRAY* state switches to the *BLACK* state and then invokes the modification protocol of Section II-A.

When node N in the *GRAY* state receives a request from neighboring site M, N determines whether $\langle$*N.gain*, *N.id*$\rangle \prec \langle$*M.gain*, *M.id*$\rangle$. If so, N performs the acceptance procedure; otherwise, message *ReqRej* is replied, causing M to fall back to the *WHITE* state. A node operating in any state other than *GRAY* simply discards arrived *ReqRej* messages.

When node M in the *RED* state receives *BlueReq* or in the *BLUE* state receives *RedReq*, both sites *M.pred* and *M.succ* are contending. Though having accepted a request, M accepts the current one if the tagged tuple indicates a higher-priority requestor. It is immaterial that both *M.pred* and *M.succ* receive *ReqAcpt*'s, since *M.pred* has sent *GreenReq* to *M.succ* where the final decision will be or has been made.

Likewise when node M in the *BLUE* state receives *GreenReq* or in the *GREEN* state receives *BlueReq*, two immediate preceding nodes are contending. Node M may accept the current request with higher priority. Site *M.pred* determines which of the two requests is actually accepted.

Given that node M in the *RED* state receives *GreenReq* or in the *GREEN* state receives *RedReq*, M is the only intersection that can arbitrate contending nodes. In this case, we specify that, if M has accepted a request, message *ReqRej* is replied in response to the subsequent request.

The above actions by node M are collectively summarized in Fig. 6. Operations not otherwise provided in the figure are as follows. Note that a requesting site in the *BLACK* state is authorized to initiate topological modifications. This site neither accepts further requests nor resets its state to *WHITE* until the intended procedure has terminated. While the modifications are progressing, node M in the *WHITE*, *GREEN*, *RED*, or *BLUE* state will receive *Set* message, thereby executing the protocol of Fig. 3. Subsequently M changes its state back to *WHITE*. The entire state transition diagram is shown in Fig. 7.

```
When (M.color = WHITE)
    On receiving RedReq$(m,n)$, BlueReq$(m,n)$, or GreenReq$(m,n)$
            from $N$
        M.gain $\leftarrow m$
        Send ReqAcpt to $N$
        Set M.color to RED, BLUE, or GREEN accordingly
When (M.color = GRAY)
    On receiving RedReq$(m,n)$, BlueReq$(m,n)$, or GreenReq$(m,n)$
            from $N$
        if ($\langle$M.gain,M.id$\rangle \prec \langle m,n\rangle$) then
            M.gain $\leftarrow m$
            Send ReqAcpt to $N$
            Set M.color to RED, BLUE, or GREEN accordingly
        else send ReqRej to $N$
When (M.color = RED)
    On receiving BlueReq$(m,n)$ from $N$
        if ($\langle$M.gain,M.id$\rangle \prec \langle m,n\rangle$) then
            M.gain $\leftarrow m$
            Send ReqAcpt to $N$
            Set M.color to BLUE
        else send ReqRej to $N$
    On receiving GreenReq$(m,n)$ from $N$
        Send ReqRej to $N$
When (M.color = BLUE)
    On receiving RedReq$(m,n)$ or GreenReq$(m,n)$ from $N$
        if ($\langle$M.gain,M.id$\rangle \prec \langle m,n\rangle$) then
            M.gain $\leftarrow m$
            Send ReqAcpt to $N$
            Set M.color to RED or GREEN accordingly
        else send ReqRej to $N$
When (M.color = GREEN)
    On receiving BlueReq$(m,n)$ from $N$
        if ($\langle$M.gain,M.id$\rangle \prec \langle m,n\rangle$) then
            M.gain $\leftarrow m$
            Send ReqAcpt to $N$
            Set M.color to BLUE
        else send ReqRej to $N$
    On receiving RedReq$(m,n)$ from $N$
        Send ReqRej to $N$
```

Fig. 6. Protocol for node M

C. Dynamic Adds and Drops

This subsection addresses how a node is added into or dropped from a ring. Procedures described here operate in a locally mutual exclusive manner as well. Therefore mechanisms of Section II-B can be applied beforehand.

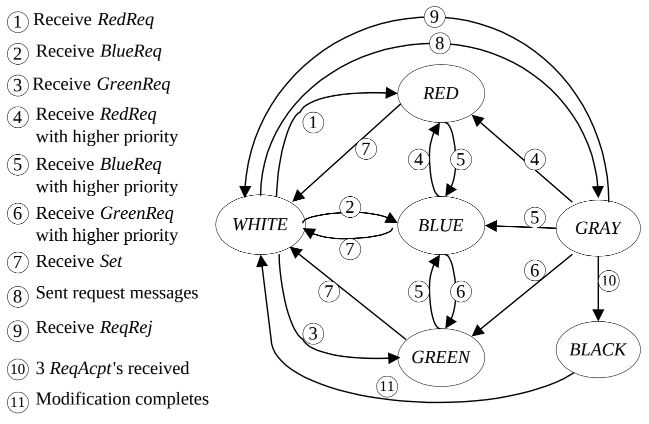
Let $\{P,Q,R,S\}$ be part of a ring. If a node N is to be inserted after Q, Q is committed to send its preceding P a message *Set*$(\_,\_,N)$ and N message *Set*(Q,*Q.succ*,*Q.succ2*), respectively. Then Q resets *Q.succ2* to R and *Q.succ* to N. On receiving the *Set* message from Q, N sends the site specified by the second parameter, namely R, another message $Set(N,\_,\_)$. Next N resumes its designated process in response to its reception.

On the contrary, to delete node N from $\{P,Q,N,R\}$, *N.pred*, viz Q, executes the actions as follows. Firstly *Q.succ* is changed to R. Then Q sends a message *Set*$(\_,\_,R)$ to its preceding P and *Set*$(Q,\_,\_)$ to R. The latter message will behoove the recipient R to hand back another *Set* message, to renew $Q.succ2$, whereupon Q instructs N to leave the computation gracefully. Henceforth N releases its internal storage space for pointers *N.pred*, *N.succ*, and *N.succ2*.

III. Correctness Proof

We now carry out the correctness proof of our mutual exclusive protocol. Here safety property refers to that two nodes in close vicinity cannot initiate structural modifications concurrently. Concerning liveness property, we show that our scheme is deadlock-free, *i.e.*, no node requesting structural modifications will be prevented indefinitely from executing its intended operations.

Lemma 1: Let P and Q be two consecutive nodes of a ring, where P precedes Q. If these two nodes request topological



1 Receive *RedReq*
2 Receive *BlueReq*
3 Receive *GreenReq*
4 Receive *RedReq* with higher priority
5 Receive *BlueReq* with higher priority
6 Receive *GreenReq* with higher priority
7 Receive *Set*
8 Sent request messages
9 Receive *ReqRej*
10 3 *ReqAcpt*'s received
11 Modification completes
RED
WHITE
BLUE
GRAY
GREEN
BLACK

Fig. 7. State transition diagram

changes concurrently, our protocol ensures that the one with higher priority can proceed.

Proof: By our protocol, P will send message *BlueReq* to Q whereas Q will send *RedReq* to P. This enables P in the *GRAY* state to learn about Q, and vice versa. Since either ⟨*P.gain*,*P.id*⟩ $\prec$ ⟨*Q.gain*,*Q.id*⟩ or ⟨*Q.gain*,*Q.id*⟩ $\prec$ ⟨*P.gain*,*P.id*⟩, only one of the two nodes will receive *ReqAcpt* from its counterpart. Hence, the node with higher priority is allowed to invoke the modification procedure. □

Lemma 2: Let $\{P, Q, R\}$ be part of a ring. If multiple nodes request topological changes concurrently, our protocol ensures that only one can proceed.

Proof: There are four possible combinations of concurrent requestors: $\{P, Q\}$, $\{Q, R\}$, $\{P, R\}$, and $\{P, Q, R\}$. The first two cases follow Lemma 1. In the third case, P will send message *GreenReq* to but receives no request from R. If ⟨*P.gain*,*P.id*⟩ $\prec$ ⟨*R.gain*,*R.id*⟩, P cannot initiate modifications, because of *ReqRej* from R. On the other hand, if ⟨*R.gain*,*R.id*⟩ $\prec$ ⟨*P.gain*,*P.id*⟩, R may still receive three *ReqAcpt*'s from other nodes. If R receives *GreenReq* from P but has not received three *ReqAcpt*'s, R will reply *ReqAcpt* to P and change its state to *GREEN*. Hence, only P is able to initiate modifications. If R receives *GreenReq* from P after having received three *ReqAcpt*'s (R in the *BLACK* state), R will reply *ReqRej* to P. At this point, only R can initiate modifications.

While P, Q, and R are contending and Q has the highest priority, P and R cannot procure the authority by Lemma 1. If Q does not have the highest priority, Q cannot secure the authority (by Lemma 1) and it suffices to consider whether both P and R can be authorized to initiate modifications. Hereinafter the argument becomes identical to that of the third case above. Reasoning shows that either P or R (but not both) can obtain the authority. □

Lemma 3: Let $\{P, Q, R, S\}$ be part of a ring. Either P or S can be authorized to initiate topological changes concurrently.

Proof: By our protocol, node R will receive *GreenReq* and *RedReq* from P and S, respectively. If P acquires the authority, (R received P's *GreenReq* first and has replied *ReqAcpt*), R will reply *ReqRej* on receiving *RedReq* from S (R in the *GREEN* state at that time). Therefore S cannot acquire the permission concurrently. If S acquires the authority, we can reason similarly that P cannot be authorized. □

Theorem 1 (safety property): Let $\{P, Q, R, S\}$ be part of a ring. If multiple nodes request topological changes concurrently, only one can proceed.

Proof: There are 11 possible combinations of concurrent requesting nodes. Three ($\{P, Q\}$, $\{Q, R\}$, and $\{R, S\}$) can be dealt with by Lemma 1, four ($\{P, R\}$, $\{Q, S\}$, $\{P, Q, R\}$, and $\{Q, R, S\}$) by Lemma 2, and one ($\{P, S\}$) by Lemma 3. The remaining three combinations, $\{P, Q, S\}$, $\{P, R, S\}$, and $\{P, Q, R, S\}$, need further discussions.

- $\{P, Q, S\}$: If P acquires the authority, neither Q (Lemma 1) nor S (Lemma 3) can be authorized. While Q acquires the authority, neither P (Lemma 1) nor S (Lemma 2) can be authorized. While S acquires the authority, neither Q (Lemma 2) nor P (Lemma 3) can be authorized. This ensures a mutually exclusive authority among P, Q, and S.
- $\{P, R, S\}$: Similar argument to the previous case can be adopted.
- $\{P, Q, R, S\}$: If either Q or R acquires the authority, this obviates other three nodes from being authorized concurrently (Lemmas 1 and 2.) If either P or S obtains the authority, this obviates other three nodes from being authorized concurrently (Lemma 3.)

□

Regarding liveness property, the following theorem indicates that our proposal is deadlock free.

Theorem 2 (liveness property): Consider a set of nodes initiating structural modifications. No node will be prevented indefinitely by others from executing its intended operations.

Proof: On the contrary, assume that there exist a set of nodes $\{N_1, N_2, N_3, \cdots,$
$N_i\}$, where $i > 1$, such that N_1 is deferred by N_2, N_2 is deferred by N_3, ..., and N_i is deferred by N_1. Such an assumption is justifiable, since a deadlock implies circular waiting. From our proposal, it follows that $\langle N_1.gain, N_1.id\rangle \prec \langle N_2.gain, N_2.id\rangle \prec \cdots \prec \langle N_i.gain, N_i.id\rangle \prec \langle N_1.gain, N_1.id\rangle$. Nonetheless this inequality can never hold, a contradiction. □

IV. Simulation Results

We conduct simulations on our protocols in the following way. Consider a ring of 100 mobile nodes which are randomly distributed over a 1000×1000 rectangle. Given any pair of nodes, we measure their physical distance as the in-between communication cost. We assume that the region under discussion accommodates intermediate hosts which, though not part of the ring, assist message delivery in the system. In other words, any two nodes in the logical structure are regarded mutually reachable throughout our simulations.

Host movements are approached by using a discrete-time based model: on each time tick, a mobile node decides its next position, at probability p that it remains stationary. Upon a movement, the heading direction is arbitrary with an exponential distribution of mean d in distance. Small d implies low mobility, for example at pedestrian walking speed. In this sequel we vary p from 0.1 to 0.9 and set d to 1, 25, and 50, respectively. If the node is to hop off the margin, it rebounds on the regional boundary. Furthermore, since our simulation model runs in a synchronous manner, a non-requesting node may receive messages *RedReq* and *GreenReq* coincidentally. Here we let such an arbitrator node always acknowledge first the *RedReq*-message source with a *ReqAcpt*, assuming that messages from the $GreenReq$ originator, which points to the arbitrator as the second successor, traverses comparatively more hops.

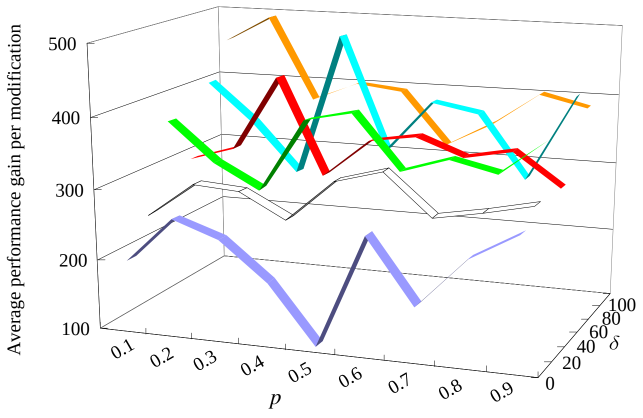
The first performance metric of interest is the total communication cost incurred by a given ring of nodes without and with applying our proposal. Corresponding measures are represented as dashed and solid lines, respectively, in Fig. 8. This figure pictures the average cost versus the probability p over 500 time units in each round. Remarkably in (a), on average our proposal has reduced the original cost of 52476.5 to the cost of 38700.2, whereas in (b) 53183.5 to 39261.8. This amounts to a saving of more than 26%, a nontrivial improvement.

In addition, we observe that p would play as a light factor to overall system performance. In case our protocol is not deployed, the mean distance d by which hosts migrate is seemingly immaterial to the incurred cost of the ring. In contrast, when our protocol is in use, the smaller d is, the more performance we tend to gain. This agrees with a commonly held belief — when mobile nodes roam less drastically, it is favorably apropos to achieve better communication efficiency by locally adjusting the ring topology, as with our strategy.

Another dimension to be examined is the average performance gain resulting from each structural modification by our protocol. On condition that d is 25, we illustrate the corresponding measures in Fig. 9, where the correlation between probability p and δ of Section II-A is considered. It can be seen that, as δ becomes larger, the benefits per modification due to applying our proposal are likely to increase. However, a larger δ will meanwhile lead to less flexible topological changes, possibly causing communication inefficiency to some extent.

V. Concluding Remarks

In mobile *ad hoc* environment, a logical structure is likely to become fragmented or costly to maintain over time. This paper presented a remedial approach to adapting a ring among mobile nodes to changeable network topology, so as to improve overall system efficiency. We modify the ring structure in a localized, mutual exclusive manner, enabling modifications to progress concurrently. Simulation results showed that our approach facilitates communication activities substantially. Our development, as a communication substrate for process arrangements, benefits ring-based distributed computations. The proposed design tenet elegantly lends itself to other settings like meshes or trees.



🟪 δ = 0 (Avg. 215.9)	◻ δ = 20 (Avg. 290.3)	🟩 δ = 40 (Avg. 345.9)
🟥 δ = 60 (Avg. 335.2)	🟦 δ = 80 (Avg. 367.6)	🟧 δ = 100 (Avg. 398.2)

Fig. 9. Average performance gain per modification versus p and δ (assuming that d is 25.)

There are pragmatic issues for future investigations. One is that, in the course of topology changes, a participant node may leave the computation due to forced termination or power off. This might result in an incorrect ring structure. To deal with, an additional mechanism needs to be introduced.

To conclude this paper, we draw two remarks on our proposal. First, a side effect might happen when more than two nodes, say N_1, N_2, and N_3, request structural modifications concurrently. N_1's request may not be fulfilled because of receiving a single *ReqRej* message from N_2, while N_2 receives *ReqRej* from higher-priority N_3. In this case, the prevention of N_1 activating modifications becomes redundant if N_1 is situated sufficiently far from N_3 (since *ReqRej* by N_2 has turned invalid and should be ruled out.) For recovery, we let N_2 which experiences request failure signal its contending neighborhood such as N_1 that received N_2-originated *ReqRej*. This revives N_1 to proceed with original request process.

Second, we tailor a ring by interchanging the positions of two logically adjacent nodes. This working paradigm, as stated previously in Section 1, is not intended for a globally optimal process organization whose maintenance requires larger scale structural adjustments. For instance, consider a scenario, where the communication cost between any two nodes is represented as their distance. Given a ring of six nodes initially configured like a regular hexagon (Fig. 10(a)), node N_4 begins moving straight towards N_1. Throughout the migration, N_4 running our protocol does not initiate any structural alteration, i.e., configuration $\{N_3, N_4, N_5, N_6\}$ does not change to $\{N_3, N_5, N_4, N_6\}$. This can easily be verified since the former configuration incurs a cost of $2d + r$, less than or equal to the latter does, $\sqrt{3}r + d + d'$, where

$$d = \sqrt{r^2 + x^2 - 2rx\cos\frac{\pi}{3}} = \sqrt{r^2 + x^2 - rx}$$

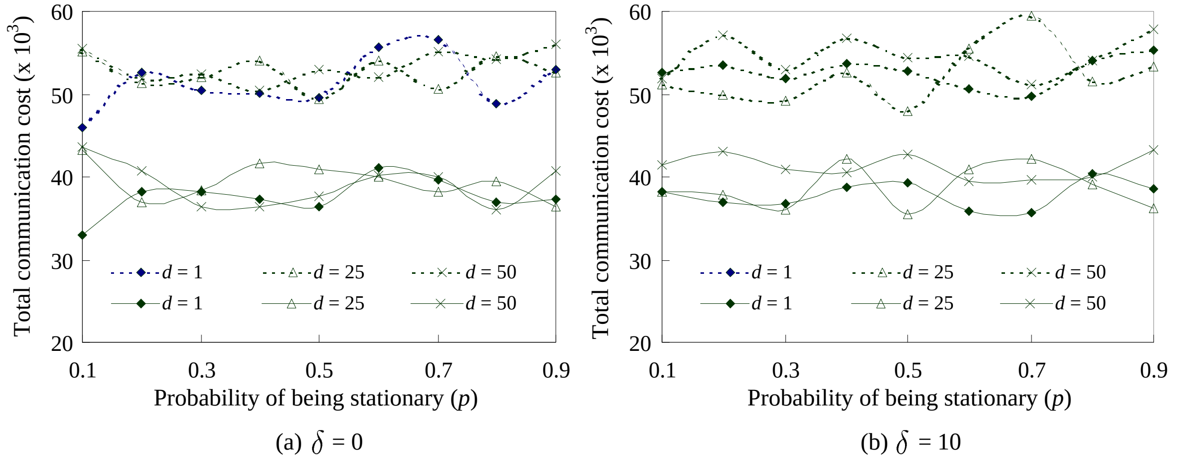


Fig. 8. Comparisons in terms of total communication cost.

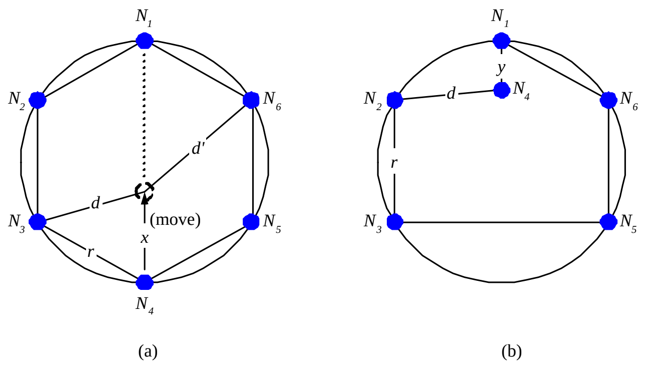


Fig. 10. Restructuring more than local modifications. (a) Original configuration. Node N_4 starts moving towards N_1. (b) A better topology when $0 < y < r/2$.

$(0 \le x \le 2r)$ and

$$\begin{aligned} d' &= \sqrt{(2r-x)^2 + r^2 - 2(2r-x)r\cos\frac{\pi}{3}} \\ &= \sqrt{3r^2 - 3rx + x^2}. \end{aligned}$$

When node N_4 reaches somewhere within $r/2$ distant from N_1, we can exploit a lower-cost structure as Fig. 10(b) demonstrates by updating connectivities among N_1, N_2, and N_4, and between N_3 and N_5. The new configuration will demand a total cost of $(3r + d + y + \sqrt{3}r)$, less than the original $6r$. However, our protocol cannot take effect in this case, because the firing condition holds only for structural modifications involving adjacent nodes. Our ring structure thus remains unchanged.

Acknowledgement

This work was supported by the National Science Council, ROC, under grant NSC 89-2213-E-216-051.

References

[1] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LAN's," in *Proc. of ACM SIGCOMM'94*, London, England, 1994, pp. 212–225.

[2] A. Iwata, C. C. Chiang, G. Pei, M. Gerla, and T. W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1369–1379, Aug. 1999.

[3] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Commun. Magazine*, pp. 46–55, Apr. 1999.

[4] E. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes," *Comm. ACM*, vol. 22, no. 5, pp. 281–283, May 1979.

[5] L. Higham and T. Przytycka, "A simple, efficient algorithm for maximum finding on rings," in *Distributed Algorithms (7th International Workshop, WDAG '93)*, A. Schiper, Ed. Lausanne, Switerland: Springer-Verlag, September 1993, pp. 249–263, also published in *Lecture Notes in Computer Science*, 725.

[6] D. S. Hirschberg and J. B. Sinclair, "Decentralized extrema-finding in circular configurations of processes," *Comm. ACM*, vol. 23, no. 11, pp. 627–628, Nov. 1980.

[7] G. L. Perterson, "An $O(n \log n)$ unidirectional distributed algorithm for the circular extrema problem," *ACM Trans. on Programming Languages and Systems*, vol. 4, no. 4, pp. 758–762, Oct. 1982.

[8] W. Jia, J. Cao, T. Y. Cheung, and X. Jia, "A multicast protocol based on a single logical ring using a virtual token and logical clocks," *Computer Journal*, vol. 42, no. 3, pp. 203–220, 1999.

[9] I. Nikolaidis and J. J. Harms, "A logical ring multicast protocol for mobile nodes," in *Proc. of 7th Int'l Conf. Network Protocols*, Toronto, Canada, Oct. 1999.

[10] A. Silberschatz, P. Galvin, and G. Gagne, *Applied Operating System Concepts*, 1st ed. John Wiley & Sons, 2000, pp. 526–527.

[11] B. R. Badrinath, A. Acharya, and T. Imielinski, "Impact of mobility on distributed computations," *Oper. Syst. Rev.*, vol. 27, no. 2, pp. 15–20, Apr. 1993.

[12] ——, "Structuring distributed algorithms for mobile hosts," in *Proceedings of the 14th International Conference on Distributed Computing Systems*, June 1994, pp. 21–28.

[13] L. M. Patnaik, A. K. Ramakrishna, and R. Muralidharan, "Distributed algorithms for mobile hosts," *IEE Proc.-Comput. Digit. Tech.*, vol. 144, no. 2, pp. 49–56, Mar. 1997.

[14] K. M. Chandy and J. Misra, "The drinking philosophers problem," *ACM Trans. on Programming Languages and Systems*, vol. 6, no. 4, pp. 632–646, 1984.

[15] F. Harary, "The maximum connectivity of a graph," *Porc. Nat. Acad. Sci.*, vol. 48, pp. 1142–1146, 1962.