간헐적 고장이 있는 완전 네트워크에서 선거 알고리즘을 위한 메시지 복잡도의 낮은 경계

김 성 동†

요 약

선거는 분산 컴퓨팅에서 기본적인 문제에 해당한다. $f \leq \lfloor (n-1)/2 \rfloor$ 일때 우리는 한 노드에 연결된 최대 f 링크를 가지는 네트워크에서 n 개의 노드를 고려한다. 보통 지도자를 선거로 뽑거나, 최대 인식자를 찾거나, 스패닝 츄리를 구성하는 것은 메시지 복잡도가 같은 차수를 가지기 때문에 동일 문제의 범주에 속한다. 본 논문은 스패닝 츄리를 이용하여 간헐 링크 고장이 있는 비동기 완전 네트워크에서 지도자 선거 알고리즘을 위한 메시지 복잡도의 낮은 경계가 $O(n^3)$ 임을 증명한다.

Lower Bound of Message Complexity for Election Alogorithm in a Complete Network with Intermittent Faults

Seong-Dong Kim†

ABSTRACT

Election is a fundamental problem in the distributed computing. We consider n nodes in the network with f maximum number of faulty links incident on each node, where $f \leq \lfloor (n-1)/2 \rfloor$. In general, electing a leader, finding the maximum identifier and constructing a spanning tree belong to the same class in the distributed computing because of the same order of the message complexity. Using a spanning tree, we prove that the lower bound of message complexity for a leader election algorithm in an asynchronous complete network with intermittent link faults is $O(n^3)$.

1. Introduction

The complete network is a network of n identical nodes except that each node has its own unique identifier from a totally ordered set. In this network, each node has $n - 1$ incident links. Communication can be executed in either a synchronous or an asynchronous mode. We consider the election problems in an asynchronous complete network. Election is a fundamental problem in the distributed computing. This problem is choosing a unique node as the leader of a network. Before starting election algorithm, no node knows the identifier of any other node. Therefore, the network cannot choose a predetermined leader. However, it is possible to elect a predetermined node as the leader of a network. In the latter case, if the leader fails or malfunctions, for instance, in the distributed file system, then it needs to choose a new

† 정 회 원 : 삼성전자주식회사 컴퓨터사업부 수석연구원
논문접수 : 1998년 7월 8일, 심사완료 : 1998년 9월 15일

leader from the rest nodes in the network in order to maintain the proper operation of whole system. The other applications of the electing leader problem contain regeneration of a lost unique token, concurrency control, recovery by electing a new central lock coordinator after crashing a coordinator in a distributed database system, and replacement a primary site in a replicated file system[1][2].

Before or during execution of the election algorithms, failures, which complicate the election problem, of nodes and communication links may occur. Fischer, Lynch and Paterson[3] proved that there is no election algorithm to solve the choosing leader problem if a node fails by crashing in an asynchronous complete network even though all the links are reliable. Hence, we consider only link failures under all reliable nodes of network. Owing to the asynchronous nature of the network, a node cannot distinguish between a slow link and a faulty link. Hence, the nodes cannot detect the faulty links. It means that the link delays are unbounded and unpredictable, and a link will not change the contents of a message at any rates. This type of fault is called *intermittent* [1][4]. If two adjacent nodes wish to communicate, then they send messages to each other on the link connecting them. If two nonadjacent nodes separated by a faulty link want to communicate, then they have to communicate through nodes adjacent to both of them.

In this paper, we prove the lower bound of message complexity in asynchronous complete networks with intermittent faulty links. Let n be the number of nodes in the network and f be the maximum number of faulty links incident on each node, where $f \leq \lfloor (n-1)/2 \rfloor$. Therefore, asynchronous intermittent faulty links in a complete network are up to $n\lceil f/2 \rceil$[4].

Because the asynchronous complete network with intermittent faults is so complicated, it is difficult to prove the exact lower bound of message complexity for electing a leader. In general, electing a leader, finding the maximum identifier and constructing a spanning tree belong to the same class in the distributed algorithms[5]. Hence, they have the same order of the message complexity. We use a spanning tree to prove the lower bound of message complexity in an asynchronous complete network with intermittent link faults.

The complete network is considered as an undirected graph $G=(P, L)$, where P is a finite set of nodes ($|P| = n$) and L is a set of links connecting two adjacent nodes, and we assume that the graph G is connected. We refer to election algorithm for a complete network as algorithm executing on the underlying graph. To prove the lower bound we count the edges which carry messages during performances of algorithms instead of the actual number of messages communicated by each edge as in [5]. In this paper, we prove that the lower bound of message complexity in an asynchronous complete network with intermittent faults is $O(n^3)$.

Many researchers have proposed several synchronous and asynchronous election algorithms for networks that have reliable nodes and links. In particular, Gafni[6] proposed an asynchronous election algorithm for *general* networks that uses $\theta(m + n\log n)$ messages, where m is total number of links in the network. For the case of asynchronous complete networks, Afek and Gafni[7], and Korach, Moran and Zaks[5] showed distributed algorithms of $\theta(n\log n)$ message complexity.

Several election algorithms have been presented for networks with faulty links. Cimet and Kumar[8] proposed an algorithm when communication links fail detectably with fail-stop mode. Goldreich and Shrira [9] presented an election algorithm in asynchronous rings with one intermittently faulty link. If n is already known to all the nodes, then their algorithm uses $\theta(n\log n)$ messages; otherwise, $\theta(n^2)$ messages. Abu-Amara[1] has studied an algorithm for election in asynchronous complete networks that uses minimally $O(nf+n\log n)$ messages, with $f \leq \lfloor n/2-3 \rfloor$. In his paper, f is the maximum total number of intermittently faulty links in the network. Abu-Amara and Lokre[4] have, in advance, develop-

ed a distributed election algorithm in asynchronous complete networks that uses $O(n^2 + nf^2)$ messages. In their algorithm, each node has at most f faulty links incident on it, where $f \leq \lfloor (n-1)/2 \rfloor$. Therefore, their algorithm introduces the tolerance up to $nf/2$ faulty links in the network. We use above mentioned network proposed by Abu-amara and Lokre, and prove the lower bound message complexity is $O(n^3)$ using spanning trees.

2. Model

Our model follows Goldreich and Shrira's model [9]. Consider an asynchronous complete network of n processors. We model the network as a complete graph on n nodes, in which each node represents a processor, and each link represents a bidirectional communication channel. Henceforth, we will not distinguish between a node and the processor it represents, and we will not distinguish between a link and the channel it represents. Each node u has a unique identifier, $ID(u)$, chosen from a totally ordered set. No node initially knows the identifier of any other node, but the nodes know that the network is complete. When a node u wishes to communicate with a node v, then u sends a message to v on the link $l(u,v)$ joining them. We assume that each message consists of at most $O(\log |T|)$ bits, where $|T|$ is the cardinality of the set of node identifiers.

A *distributed algorithm* on the network is a set of n deterministic local programs, each assigned to a node. Each local program consists of *computation statements* and *communication statements*. The *computation statements* control the internal computation of a node. The *communication statements* are of the form "send message M on link l" or "receive message M' on link l". Each node u has a Send-Buffer(u,l) and a Receive-Buffer(u,l) associated with each link l incident on u, where the buffer is not necessarily first-in first-out. Let l be $l(u,v)$. When u wishes to send message M on l, u places M in Send-Buffer(u,l). We call this event a *send event*. To capture the asynchronous nature of our network, messages may remain in the send-buffers for arbitrary lengths of time. A *transmission event* in l occurs when l places M in Receive-Buffer(v,l). We assume that u cannot inspect Send-Buffer(u,l) to check whether M was removed from the buffer. Hence, M is *in transit* from u to v if M is in Send-Buffer(u,l). If u wishes to process a message M' from Receive-Buffer(u,l). We call this event a *receive event*. For convenience, we assume that it takes one time unit to remove M' from Receive-Buffer(u,l) and to execute the computation statements on M'. If M' is not in Receive-Buffer(u,l), then u either waits for M', or receives some other message, depending on u's local program. Note that when we say that node u receives a message, we mean that u *removes* the message from a Receive-Buffer and *processes* the message. A *loss event* in a link l is the event of l discarding a message.

A link is *faulty* during or before a particular execution of the algorithm if l experiences at least one loss event in the execution of the algorithm. Links may fail intermittently at any time during the execution of the algorithm. If a link l is not faulty, then it is *reliable*. A reliable link is a link that never loses messages. Recall that each node is adjacent to at most f faulty links. Therefore, if l is faulty, and l connects nodes u and v, both u and v can be adjacent to at most $f - 1$ faulty links in addition to a faulty link l.

All the nodes in the network are reliable. It is not necessary for all nodes to start the execution of the algorithm simultaneously; some node may be initially asleep. We assume that, if a dormant node receives a message from some other node, then it wakes up and processes the received message. The processing time at each node is assumed to be negligible compared with link delays. There may be two statuses of a node during execution of the algorithm, *live* or *dead*. Initially, all the nodes are *live*. If a node u finds that there is a larger $ID(v)$ than $ID(u)$ at some time t, then u is *dead* at t. A

dead node cannot participate in the competition for the leader. However, a dead node may send messages to some nodes after receiving messages from other nodes. If there is only one live node, a unique node is elected as a leader of the network.

Consider a particular execution E of a distributed computing. Let $Events(E)$ be the multiset of the events in E. For convenience, we assume the existence of a global clock that gives the time at which each event in E occurs. Although this clock is available to an observer of the network, the nodes do not know of its existence. We will assume that each event in E occurs at some discrete unit of time starting from zero. Let $Events(u)$ be the multiset of u's send and receive events in E. The local program in u induces a total ordering on $Events(u)$. Two events, each in a distinct node, may occur at the same time. However, two events cannot occur at the same time in the same node.

In this paper, we follows Korach, Moran and Zaks's definitions[5] to prove the lower bound of message complexity. The communication network is viewed as an undirected graph $G=(P, L)$ with $|P|=n$, and we assume that the graph G is connected. We refer to an algorithm for a given network as the algorithm on the underlying graph.

E is acting on a graph $G=(P, L)$. With each execution we can associate a sequence $SEND=\langle send_1, send_2, ..., send_k\rangle$ that includes all the events in their order of occurrence. We identify each event $send_i$ with the 4-tuple $(p_{origin}(send_i), p_{forward}(send_i), l(send_i), m_i)$, where $p_{forward}(send_i)$ is the node sending the *forward* message m_i originated by $p_{origin}(send_i)$ on the link $l(send_i)$. If the two live nodes communicate each other directly, we represent the event such as $(p_{origin}(send_i), \phi, l(send_i), m_i)$. $send_i$ occurred at time t_i, where $t_i > t_{i-1}$ for $i > 1$.

Let $SEND(t)$ be the prefix of length t of the sequence $SEND$, namely $SEND(t)=\langle send_1, ..., send_t\rangle$. If $t < t'$ then we say that $SEND(t')$ is an extension of $SEND(t)$,and we denote $SEND(t) < SEND(t')$. $SEND$ is called a completion of $SEND(t)$.

Let $NEW=NEW(SEND)$ be the subsequence $\langle new_1, new_2, ..., new_r\rangle$ of the sequence $SEND$ that consists of all the events in $SEND$ that use previously unused links. A link is used if a message has already been sent along it from either side. This means that the message $send_i=(p_{origin}(send_i), p_{forward}(send_i), l(send_i), m_i)$ belongs to NEW if and only if $l(send_i)\neq l(send_j)$ for all $j < i$. $NEW(t)$ denotes the prefix of size t of the sequence NEW. Define the graph $G(NEW(t))=(P, L(NEW(t)))$, where $L(NEW(t))$ is the set of links used in $NEW(t)$, and call it the graph *induced* by the sequence $NEW(t)$. The link complexity $l(E)$ of E acting on a graph G is the maximal length of a sequence NEW over all executions. The message complexity $m(E)$ if E acting on a graph G is the maximal length of a sequence $SEND$ over all executions. It is clear that $m(E)\geq l(E)$.

3. The lower bound

Before proving the lower bound in our model, we show that it is impossible for the lower bound to be $O(n^2)$.

Claim 1. Why does a dead node send *forward* messages?

We choose arbitrarily two live nodes, p_i and p_j, $i < j$. At first, the node i sends messages to all neighbors. If it receives $\lfloor(n-1)/2\rfloor$ successful replies from other nodes which have smaller ID than $ID(i)$, the node will be a candidate for a leader, and all nodes that sent successful replies to node i are dead. Next, the node j also sends messages to all its neighbors. Eventually it enters candidate state like the node i. In the worst case, we assume the status of all links is unchanged. If there are n live nodes, there can be $(\lfloor(n-1)/2\rfloor+1)$ candidate state nodes at most. Due to the unchanged status of all links, the algorithm cannot elect the leader. It means the network is in deadlock. Therefore, when dead

nodes that were already visited by several nodes receive messages from the largest *ID* node ever visited, these dead nodes have to send forward messages to the visited nodes to compete live nodes each other. □

Claim 2. Why is not the lower bound of message complexity $O(n^2)$?

We assume there are n processors, $p_1 < p_2 < \cdots < p_n$. We define that the suppressor is the largest *ID* node that ever visited to the arbitrary dead node. When a dead node receives a message from the largest *ID* node visited on it, if that node sends a forward message only to his suppressor, the lower bound of message complexity is $O(n^2)$. For example, we assume p_1 is dead and p_2, p_3, and p_4 are live nodes. Eventually node p_2 and p_3 will be dead, sometimes through the dead node p_1, and p_4 will recognize the status of p_3. However, p_4 cannot know the status of p_2. Therefore, we see that $O(n^2)$ is not sufficient to elect a leader in our model. □

We show above that it needs forward messages and the lower bound of message complexity may be larger than $O(n^2)$ in an asynchronous complete network with intermittent faults. From now on, let us prove the lower bound. We assume that there are n identical nodes in the complete asynchronous networks with maximally $(n/2)$ $\lfloor (n-1)/2 \rfloor$ intermittent faulty links. These identical nodes but different identifier are $p_1 < p_2 < \cdots < p_n$. Under this model, we look at the link complexity in the worst case. Initially all nodes are in live state and start a distributed algorithm separately.

To be the worst case, the smallest *ID* node is dead at first, and the second smallest *ID* node is dead next, finally there is one live node, the largest *ID* node, because of eliminating the smallest *ID* node in the live nodes one-by-one. When the smallest $ID(p_1)$ competes one another node except the second smallest $ID(p_2)$, p_1 will be dead. This situation has a fewer number of messages than the case that p_2 is the suppressor of p_1. For example, when p_3 is the suppressor of p_1, since p_2 cannot construct a spanning tree, p_1 is dead by p_2 and p_2 is dead by p_3 to get more link complexity.

We let n be $2k + 1$. We consider that k is the maximum number of faulty links. After the k smallest *ID* nodes are dead, p_{k+1} node has k son nodes. The live nodes can compete each other through their sons. If a node is dead, then it has to inform that its status is changed. Also the sons of the dead node must send the result to their new suppressor. At this time, the dead node and its sons communicate through new links.

Therefore, the maximum number of link complexity to construct a spanning tree is as followings;

(1) All nodes broadcast when the distributed algorithm starts. At first each node which has $n-1$ incident links acts as a root of a spanning tree. The order of message complexity is $O(n^2)$.

(2) The dead nodes send forward messages to construct a spanning tree. We assume that p_1 ,p_2, p_3, ... is dead respectively. When p_1 is dead by p_2, there is no forward messages. After that, when the dead node p_1 receives a message originated from the live node p_3, it sends a forward message to the live node p_2 to compete both p_2 and p_3. p_2 would be dead by p_3. Next, the messages originated from p_4 are arrived at the dead nodes, p_1 and p_2. p_1 sends the forward messages to p_2 and p_3, and p_2 sends the forward message to p_3. Eventually p_n will remain as an only live node.

$$1+(1+2)+(1+2+3)+\cdots+(1+2+3+\cdots+(k-2))+(1+2+3+\cdots+(k-2)+(k-1))+(1+2+3+\cdots+(k-2)+(k-1)+k)+(1+2+3+\cdots+(k-2)+(k-1)+k)+(2+3+4+\cdots+(k-1)+k+(k+1))+\cdots+(k+\cdots+(k+(k-2))+(k+(k-1))$$
$$= (1/6)(7k^3 - k) = O(n^3).$$

(3) When one node is dead, this node informs of its sons to change their suppressor.

$1+2+\cdots+(n-2) = O(n^2)$.

Therefore, totally link complexity $I(E) = n(n-1) + O(n^3) + O(n^2)$.

We conclude that the lower bound of message complexity is $O(n^3)$, for a leader election algorithm in an asynchronous complete network with intermittent link faults using a spanning tree.

4. Conclusion

Election is the problem of choosing a unique processor as the leader of a network of processors. No node knows the identifier of any other node at first phase. The network cannot thus elect a predetermined leader. The processors do not have shared memory and can communicate with each other only by sending messages to each other on the communication links connecting them. We consider election algorithm in an asynchronous complete of n identical nodes except that each node has its own unique identifier from a totally ordered set. Hence, a node cannot distinguish between a slow link and a faulty link because of the asynchronous nature of the network. We call this type of link failues the intermittent fault. Let f be the maximum number of faulty links incident on each node, where $f \leq \lfloor (n-1)/2 \rfloor$. Thus asynchronous intermittent faulty links in a complete network are up to $nf/2$. In this paper, after showing that it is impossible for the lower bound to be $O(n^2)$ we proved that the lower bound of message complexity in an asynchronous complete network with intermittent link failures is $O(n^3)$, using a spanning tree.

References

[1] H. H. Abu-Amara, "Fault-tolerant distributed algorithm for election in complete networks," *IEEE Transactions on Computers*, Vol.37, pp. 449-453, April 1988.

[2] D. A. Menasce, G. J. Popek, and R. R. Muntz, "A locking protocol for resource coordination in distributed databases," *ACM Tran. Database Syst.*, Vol.5, pp.103-138, June 1980.

[3] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. Ass. Comput. Mach.*, Vol.32, pp.374-382, April 1985.

[4] H. H. Abu-Amara and J. Lokre, "Election in complete asynchronous networks with intermittent link failures," *IEEE Transactions on Computers*, Vol.43, pp.778-788, July 1994.

[5] E. Korach, S. Moran, and S. Zaks, "Tight lower and upper bounds for some distributed algorithms for a complete network of processors," *Proc. 3rd ACM Symp. Principles Distributed Comput.*, Vancouver, B.C., Canada, pp.199-207, Aug. 1984.

[6] E. Gafni, "Improvements in the complexity of two message-optimal election algorithms," *Proc. 4th ACM Symp. Principles Distributed Comput.*, Minacki, Ont., Canada, pp.175-185, Aug. 1985.

[7] Y. Afek and E. Gafni, "Time and message bounds for election in synchronous and asynchronous complete networks," *Proc. 4th ACM Symp. Principles Distributed Comput.*, Minacki, Ont., Canada, pp.186-195, Aug. 1985.

[8] A. Cimet and P. R. S. Kumar, "A resilient distributed protocol for network synchronization," *ACM SIGCOMM Symp. Commun. Arch. Protocols*, Stowe, VT, pp.358-367, Aug. 1986.

[9] Goldreich and L. Shrira, "The effect of link failures on computations in asynchronous rings," *Proc. 5th Symp. Principles Distributed Comput.*, Calgary, Alta., Canada, pp.174-185, Aug. 1986.

김 성 동

sdkim@halla.sec.samsung.co.kr

1983년 경북대학교 전자공학과 졸업(공학사)

1990년 경북대학교 전자공학과 졸업(공학석사)

1996년 미국 텍사스 A&M 대학교 전자공학과 졸업(공학박사)

1983년 3월~1988년 9월 한국전자통신연구소 공정제어 연구실 연구원

1996년 7월~현재 삼성전자주식회사 컴퓨터사업부 수석연구원

관심분야 : 컴퓨터구조, 컴퓨터네트워크, 병렬 및 분산처리