Zone-Based Self-Organized Clustering with Byzantine Agreement in MANET

Soonhwa Sung

Abstract: **The proposed zone-based self-organized clustering broadcasts neighbor information to only a zone with the same ID. Besides, the zone-based self-organized clustering with unique IDs can communicate securely even if the state transition of nodes in zone-based self-organized clustering is threatened by corrupted nodes. For this security, the Byzantine agreement protocol with proactive asynchronous verifiable secret sharing (AVSS) is considered. As a result of simulation, an efficiency and a security of the proposed clustering are better than those of a traditional clustering. Therefore, this paper describes a new and extended self-organized clustering that securely seeks to minimize the interference in mobile ad hoc networks (MANETs).**

Index Terms: **Byzantine agreement, cluster head, proactive asynchronous verifiable secret sharing (AVSS), self-organized clustering, zone-based self-organized clustering, zone building with IDs.**

I. INTRODUCTION

In organisms of the natural world, there are diverse methods to survive from their enemy as follows: Changing body colors, peculiar smell, transforming body form, and so on. They voluntarily find their solutions to survive from their enemy. In networks of the communication, similarly, many researchers had argued about survival from their enemy. Especially, the methods of coping with obstructions of a communication in mobile ad hoc network (MANET) are different from that of a traditionary communication because MANET is a kind of wireless ad hoc network, and is a self-configuring network of mobile routers and associated hosts connected by wireless links.

These limitations of MANET need to communicate for user requirements effectively and securely. For the efficiency of MANET, there are two studies on organizing mobile nodes. One study is fully distributed network where each station takes the role of both an ordinary node and a gateway. The other study is a hierarchical architecture in which some nodes are picked as cluster heads or gateway nodes. For a security of MANET, some research challenges such as securing the communication link, securing distributed services, tolerating captured nodes.

In this paper, the algorithm for self-organized nodes which tolerates corrupted nodes in MANET is considered. For an efficiency, communications of MANET consider a zone-based self-organized clustering that does not broadcast neighbor information to the entire network and extends the self-organized clustering algorithm to deal with network mobility. The self-organized clustering algorithm deletes nodes from clusters after nodes move away or reforming clusters after cluster heads move out of the clusters. For the security, the idea of tolerating corrupted nodes in the proposed algorithm describes Byzantine agreement protocol of proactive asynchronous verifiable secret sharing (AVSS).

This paper is organized as follows: Self-organized clustering algorithm as related work in Section II, zone-based clustering in Section III, Byzantine agreement protocol with proactive AVSS in Section IV, the proposed zone-based self-organized clustering algorithm in Section V, simulation results in Section VI, conclusions in Section VII.

II. RELATED WORK (SELF-ORGANIZED CLUSTERING ALGORITHM)

A cluster is a group of loosely coupled nodes that work closely together so that in many respects they can be viewed as though they are a single node in networks. The self-organized clustering algorithm is a distributed algorithm and each node executes the algorithm independently and the failure of one node does not affect the rest of the network. The goal of the algorithm is to minimize the summation of interference generated by all clusters, subject to the constraints that 1) all nodes within a cluster head's radio range should be considered as potential nodes in the cluster, and 2) there should be at most N_u nodes in a cluster excluding the cluster head. N_u is calculated based on the result from [1] in which the maximum number of nodes that can be supported in a cluster for a given data rate is determined. Different values can be chosen to satisfy different data rate requirements. The first constraint prevents the size of a cluster from getting so small that the network degrades to a fully distributed network. The second constraint prevents a cluster from getting so large that the data rates have to be reduced to an undesirable level.

Because the cluster head is the dominant power emitter in a cluster, the total transmission power of the cluster heads has to be minimized to reduce the interference generated by the clusters. Interference emitted by the ith cluster head can be defined as the total transmission power emitted from the cluster head expressed as the summation of the transmission power in each downlink physical channel (indexed by k) in the cluster (indexed by i)

$$\sum_{k=1}^{M} P_{ik}. \tag{1}$$

Here, P_{ik} is the transmission power needed for the ith cluster head to reach the kth node in the cluster and N_i is the number of nodes controlled by the ith cluster head. Interference can be further expressed as follows, assuming perfect power control.

$$\sum_{k=1}^{M} D_{ik}^{n} C. \tag{2}$$

Manuscript received November 15, 2007.

The author is with the Department of Computer Engineering, Chungnam National University, Korea, email: shsung@cnu.ac.kr.

Here, n is the radio propagation path loss factor, D_{ik} is the distance between the ith cluster head and the kth node in the cluster, and C is a constant that relates to a signal's center frequency, transmit and receive antenna gains, and the received power.

The interference factor for the ith cluster head is then defined as follows. Since C is the same for all the nodes in the network assuming perfect power control and homogenous nodes, C is dropped from (3)

$$I_i = \frac{1}{N} \sum_{k=1}^{N_i} D_{ik}^n. \tag{3}$$

Note that the node with the smallest interference factor is the node closest to its neighbors.

If we define the interference factor of a cluster to be the product of a cluster head's interference factors, I_i, and the number of nodes in the cluster, N_i, the objective of the self-organized clustering algorithm is to minimize the summation of the interference factors of all clusters

$$\min \sum_{i=1}^{M} N_i I_i = \min \sum_{i=1}^{M} \sum_{k=1}^{N_i} D_{ik}^n \tag{4}$$

subject to the following constraints.

1) All nodes within a cluster head's radio range are considered as potential nodes in the cluster, and
2) $N_i \leq N_u$, for $i = 1, \cdots, M$. M is the total number of clusters in a network. The first constraint ensures that a cluster head does not reject a node's connection when N_i is less than N_u [2].

III. ZONE-BASED CLUSTERING

A zone is a set of nodes that are located close to each other. Clusters are built within each zone not allowing inter-zone clusters. Node sharing between clusters is prohibited to prevent duplicate nodes. Zoning process is distributed. Seed nodes that are located at the boundary of the network announce themselves as initiators and start to build zones at the same time.

In the beginning all nodes are themselves zones, a one-member zone. A node can be in several states; initially all nodes are SLEEP state as shown Fig. 1. A zone must have a coordinator which determines whether to merge or not with neighbor zones. A coordinator that actively starts the zone-merging process is called an initiator. So, zoning process is a negotiation process between two coordinators that represent two neighboring zones: One of them is an initiator, the other a participant. The state changes of these two nodes are shown in Fig. 1 [3].

When zoning process begins, the seed nodes send *setup_init* messages to their neighbors. The seed nodes in this case act as initiators, and the neighbors become participant. The participants send back the size of the zone they belong to. The initiator examines the size and determines whether to merge or not. The basic criterion is zone size after merging; if it becomes greater than some threshold (t), the merging is abandoned.

A. Zone-Based Cluster Formation

The proposed work considers reforming clusters after cluster heads move out of the clusters. To reform a self-organizing cluster scenario when the cluster head moves out of the cluster in a self-organizing of MANET clustering, this work would like to research a self-organizing cluster head position.

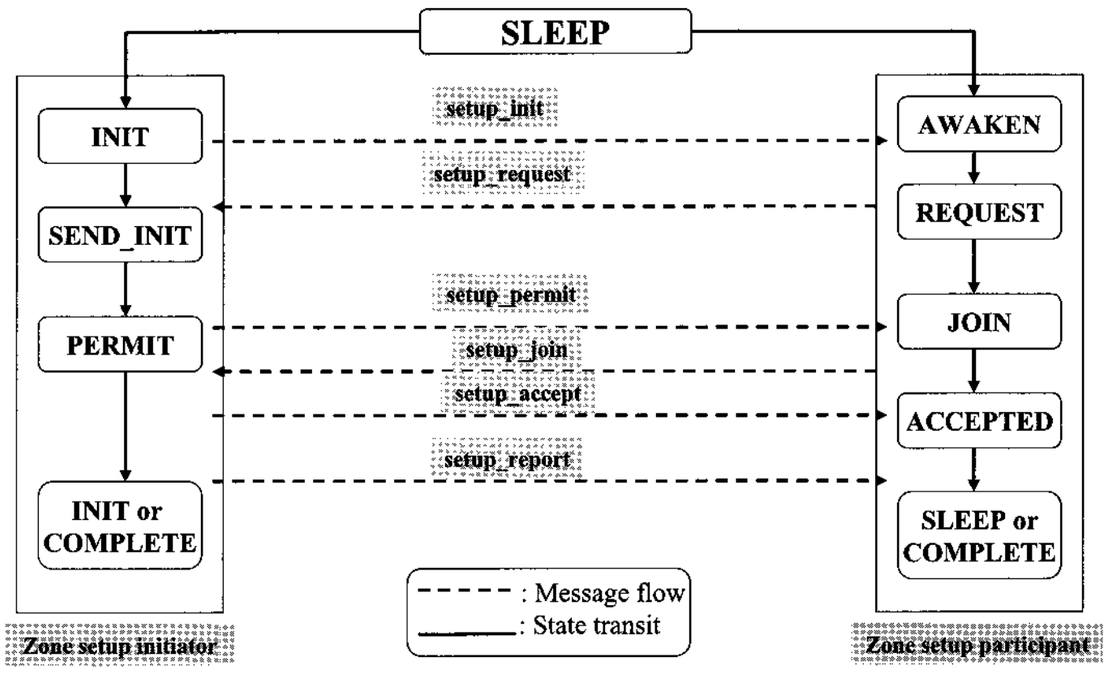

Fig. 1. State transition of a node.

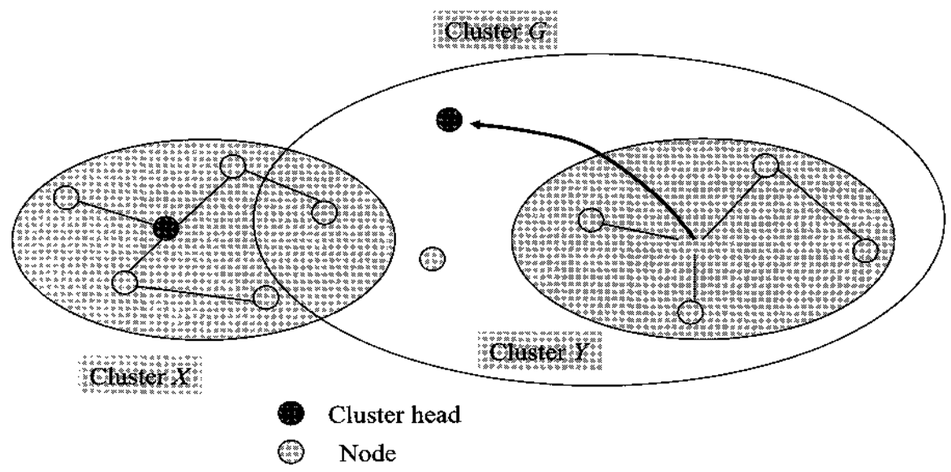

Fig. 2. Cluster head out of a cluster.

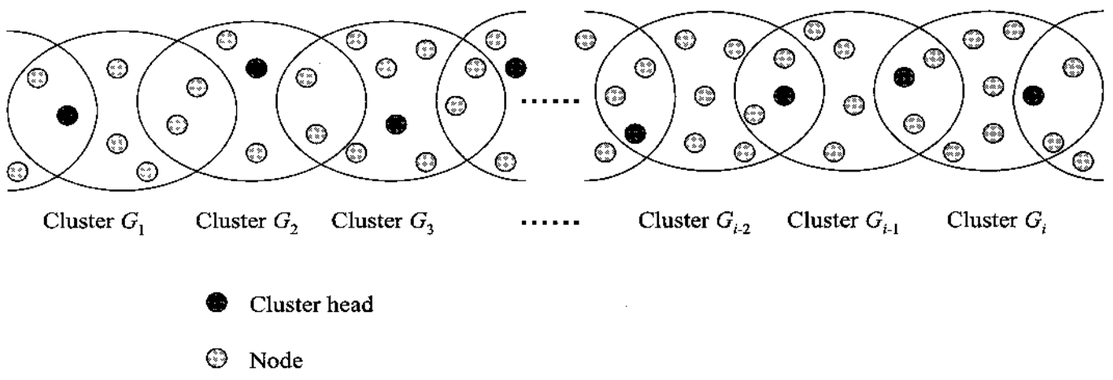

Fig. 3. Cluster head position in new-reformed cluster.

Fig. 2 depicts a self-organizing cluster head position when the cluster head moves out of the cluster. When the cluster Y head moves out of cluster Y, MANET's nodes reform cluster G not including the cluster X head. That is, new-generated cluster which includes bolted cluster head has only one cluster head.

These sequential cluster reformations don't include any cluster head which already has included in new-reformed cluster in Fig. 3 [4].

Cluster G_1 has a cluster head such as Fig. 3, but cluster G_2 doesn't have the same cluster head in G_1 even if G_1 cluster head moves out of cluster G_1. The basic criterion is cluster size after reforming; if it becomes greater than some threshold (t), the reforming is abandoned.

In cluster G_i, cluster head has unique node ID which has included the secret key on behalf of cluster G_i. The cluster head position in new-reformed cluster is applied to zone-based clustering, and each nodes of cluster G_i share the secret key of cluster head ID. The process is omitted in this work.

B. Secure Zone Building with IDs

Merging decision may be made for several neighbor zones at the same time. The *setup_init* messages are sent to all neighbor coordinators, and if multiple responses arrive within the same time period, the initiator would consider all the corresponding participants as possible candidates for merging. The response from the participant is a *setup_request* message, and it contains the responder's zone size. The initiator uses this information to give preference to smaller participants: It is again to deter the formation of fragmented clusters. The participant does a similar selection based on the size of a zone. The *setup_init* message also contains the zone size of the initiator, and when the participant has received a number of *setup_init* messages, it prefers the initiator with the smallest zone size and sends *setup_request* message to it [3].

When a new zone is formed through merging, a new coordinator has to be elected. For this purpose, the old coordinator broadcasts a *coordinator_election_request* message to all zones with unique ID. Each zone responds with the neighbor zone table it has. The old coordinator counts only the neighbor nodes that do not belong to the current zone, and the zone with the largest neighbor node number (excluding ones within the current zone) will be elected as the new coordinator. The node number of the new coordinator will be notified to all zones, and this new coordinator will start the next phase of zone merging process.

Merging process stops when all the neighbor zones are merged to the initiator's zone or when the size of zone has reached the threshold (t). When the threshold (t) size is reached, the process has a final zone, and the state of all nodes within this zone becomes complete. When the threshold (t) is not yet reached, the process repeats a zone merging again.

Above mentioned cluster heads have their IDs and the total transmission power of the cluster heads has to be minimized to reduce the interference generated by the clusters. But a self-organized clustering algorithm must broadcast neighbor information to the the entire network. Not to broadcast neighbor information to the entire network, a zone-based self-organized clustering is considered. Zone building with IDs is dependent on its character. Its character is composed of various requirements of a user in MANET. The requirements of a user make a kind of zone building with IDs which has collected some nodes in MANET.

If a zone is reformed in MANET, a zone has unique ID without transmission of neighbor information to the the entire network. Due to unique ID of a zone, a zone only broadcasts informations in the same zone to the other zone.

Zone building with IDs has the size of zone with some threshold (t). The threshold (t) changes in order for the organisms of the natural world to survive from their enemy whenever zone building with IDs is attacked by adversaries. That is, communications to decide threshold (t) have Byzantine agreement protocol in Section IV.

Zone IDs delegates cluster head IDs, and cluster head IDs have hierarchical construction. The construction is not described in this work.

When the seed nodes are threatened by corrupted nodes, zoning process does not begin. That is, the seed nodes do not send *setup_init* messages to their neighbors, so a station transition

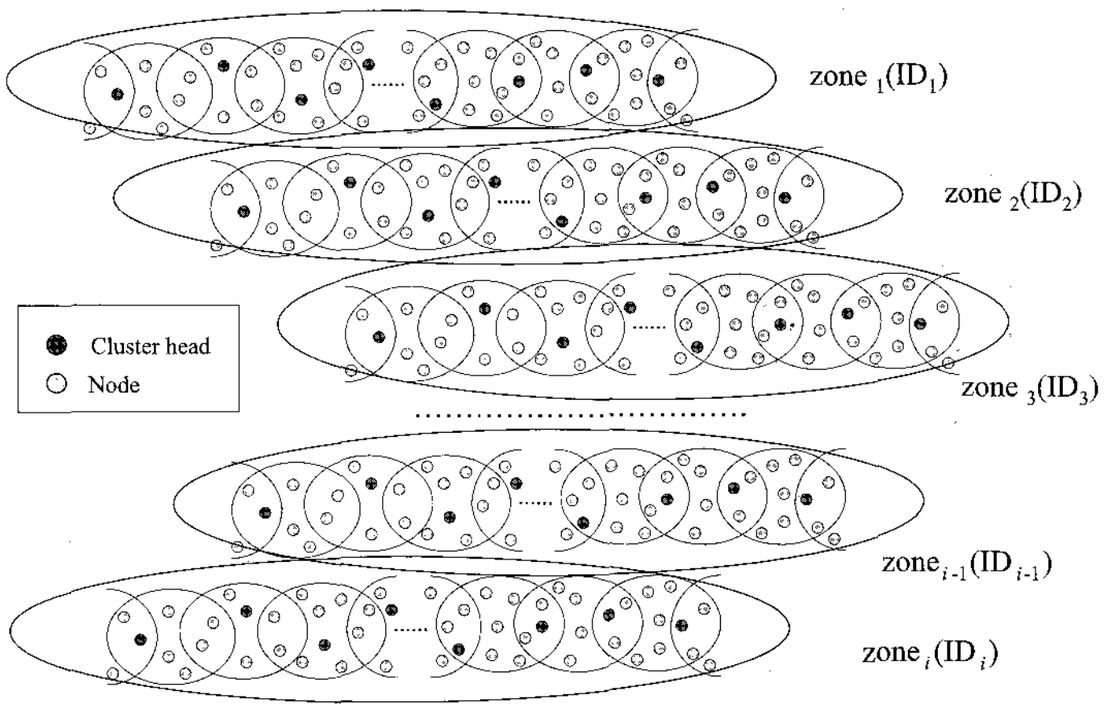

Fig. 4. Zone building with IDs.

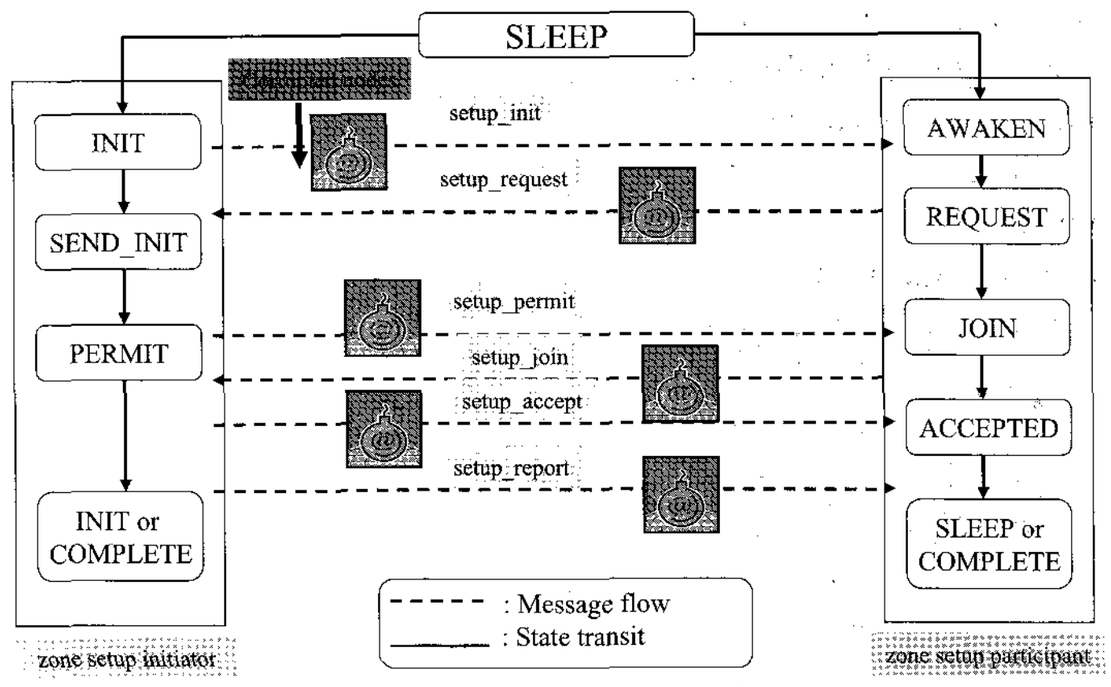

Fig. 5. State transition of a corrupted node.

of a node does not get processed. *Setup_request* messages do not transit zone setup initiator, and *setup_permit* messages do not send zone setup participant. Besides, *setup_join* messages, *setup_accept* messages, and *setup_report* messages do not proceed state transition any more. Therefore, this work needs the protocol so that it can communicate securely even if the state transition of nodes in zone-based clustering is threatened by corrupted nodes.

Proactive secret sharing allows to refresh all shares by generating a new set of shares for the same secret key from the old shares without reconstructing the secret key, and then the old share is useless after the refresh of each share. Thus, proactive secret sharing reasonably provides the way to escape from threats of exposing the secret key when threshold cryptography is used. Therefore, this work in zoning process considers the Byzantine agreement protocol with proactive AVSS.

IV. BYZANTINE AGREEMENT PROTOCOL WITH PROACTIVE AVSS

A. Preliminaries

The network node of large networks, such as a sensor network, is usually unknown, since a sensor network is always dynamically changing as follows: New nodes are always added to the network, while some nodes go down. But the traditional

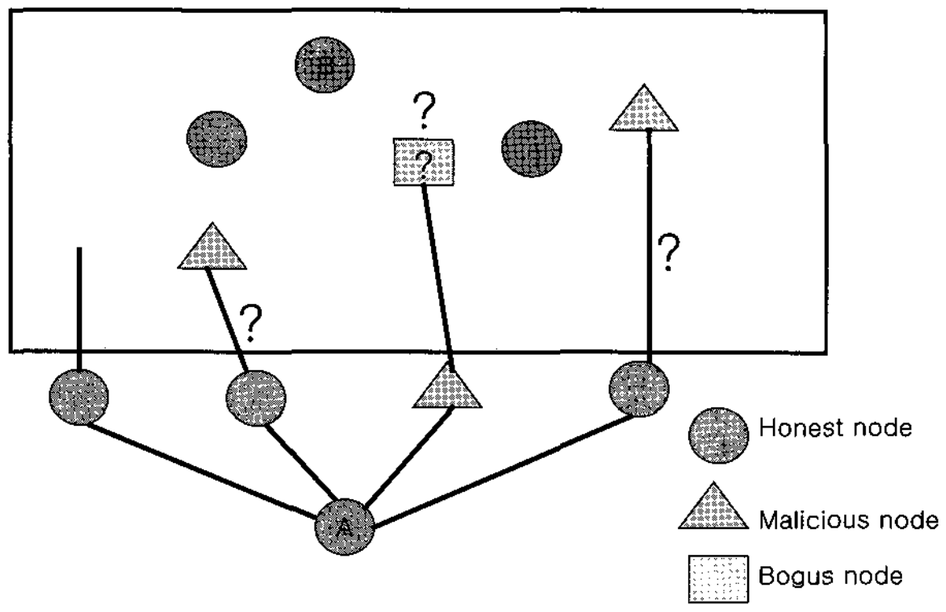

Fig. 6. The nodes of wireless networks.

work on Byzantine faults [5], [6] covers only the case that the network is known. When the network is unknown faulty nodes not only send faulty data, but these can also pretend that bogus nodes and bogus edges exist. Therefore, the scheme is focused on developing scheme to deal with Byzantine faults in sensor network without a trusted central certification authority. Because a new node joining the wireless sensor network needs to efficiently and autonomously set up secret keys with his communication partners without the use of a central infrastructure.

The scheme considers to control over corrupted parties, especially, with Byzantine adversaries that may alter the behavior of the corrupted parties in an arbitrary and coordinated way. In this Fig. 6 [7], the node A wants to communicate with node B. However, it only knows its neighbors, and does not know the outside network structure. Some nodes may be malicious. In addition, since nodes are severely energy constrained and delete infeasible to replace the batteries of thousands of nodes, the key challenge in networks is to maximize the lifetime of nodes. Therefore, securing and computation operations of nodes, and communication protocols must be made as efficient as possible.

B. Byzantine Agreement Perspectives

The Byzantine generals problem was first proposed by Lamport, Shostak, and Pease [5], in ACM Transactions on Programming Languages and Systems 4.3. Their concern was how to create a consensus among a number of independent processes, when some of those processes might be faulty, deadlocked, or fail to send correct messages. However, since the problem hinges around trust, and since cryptography is, largely, about trust, it is a significant problem in cryptography as well.

Consider this problem: A city is surrounded by the Byzantine generals, and their troops. The generals must come to an agreement about whether to attack or retreat. But the rulers of the city may have bribed some of the generals to disrupt their procedure for coming to a consensus, and no loyal general knows which other generals have been bribed (or, indeed, whether any other generals have been bribed at all). What the rulers of the city want is for some of the generals to attack and some to retreat—they can easily defeat a divided force.

It is crucial that all loyal generals agree on the same course of action: If some were to advance while others retreated, they would be annihilated. They need a secure method to come to such an agreement, so that they can know that each and every one of them has come to the same agreement.

What they need is called a *Byzantine agreement*. The properties of a Byzantine agreement are

- every actor who executes it correctly will have the same information about what every other actor said.
- If an actor executes it correctly, then all other actors will have the same true information about what that actor said.
- Every actor will be certain that all the other actors have the same information that s/he has.

There are protocols for Byzantine agreement, reflecting different conditions. If each disrupting general is capable of lying about what another general said, then if one-third or more of the generals disrupt the protocol, no consensus is possible. In the case where the generals cannot lie about what message they received from another general (via use of digital signatures or similar) then a Byzantine agreement is always possible as long as there are at least two loyal generals to come to such an agreement.

Byzantine agreement protocols imply large communication overhead due to creating a consensus among a number of independent processes in fault and deadlock of communications.

On the other hand, ad hoc networks increasingly rely on fault-tolerant and secure authorization services for seamless network services. An essential primitive used to implement such services is the Byzantine agreement protocol for achieving agreement among n parties even if t parties $(t > n/3)$ are corrupt and behave maliciously. Therefore, this work proposes a Byzantine agreement protocol.

C. Byzantine Agreement Protocol

The scheme uses the following Byzantine adversaries that may alter the behavior of the corrupted parties in an arbitrary and coordinated way. A standard formulation of the Byzantine agreement problem is as follows: Design a protocol that allows the corrupted parties to agree on a common value. The agreed value should be the input value of one of the uncorrupted parties.

Definition: Let π be an n-party protocol where each party has a binary input. Protocol π is a $(1-\varepsilon)$-terminating, t-resilient Byzantine agreement protocol if the following requirements hold, for every t-adversary and every input.

- Termination: With probability $1-\varepsilon$, all the uncorrupted parties complete the protocol (i.e., terminate locally)

- Correctness: All the uncorrupted parties who have terminated have identical outputs. Furthermore, if all the uncorrupted parties have the same input, σ, then all the uncorrupted parties have output σ [8].

Let A be a set of parties. Let the conversation, C_A, among the parties in A, be the random variable having the distribution of the sequence of all messages sent among the parties in A. The order of the messages in each instance of the conversation is induced by their order in the corresponding view.

Theorem 1: For every $n \geq 4$, there exists functions f such that no asynchronous protocol for n parties securely $\lceil \frac{n}{4} \rceil$-computes f, if Byzantine adversaries are allowed [8].

Theorem 2: Let $n \geq 3t+1$. For every $\varepsilon > 0$, there exists a $(1-\varepsilon)$-terminating, t-resilient, asynchronous Byzantine agreement protocol for n parties. Conditioned on the event that the

honest parties terminate, they do so in constant expected time. Furthermore, the computational resources required of each party are polynomial in n and $\log 1/\varepsilon$ [8].

The work considers to control over corrupted parties, especially, with Byzantine adversaries that may alter the behavior of the corrupted parties in an arbitrary and coordinated way [7].

D. Basics of Threshold Cryptography

A cryptosystem corresponds to evaluate a function with two inputs [9]. One of them is the key, and the other one varies from application to application, with examples being: The plaintext, the ciphertext, the text to sign, a seed, etc. Often this function is written as having one input with a key as parameter, e.g., $f_{\text{key}}(\text{input})$. In this context it is useful to view that input as a parameter, so that we have

$$f_{\text{key}}(\text{input}) = g_{\text{input}}(\text{key}) \tag{5}$$

where f and g are functions.

Some important cryptographic schemes have the property where a function g, playing a major component, is homomorphic [10], i.e.,

$$g_b(k_1 + k_2) = g_b(k_1) * g_b(k_2) \tag{6}$$

where b is the aforementioned input, and k_1, k_2 belong to the key space.

As a first example consider the ElGamal encryption scheme [11]. First the public key is (g, y, p) where g has a large enough order, p is a large enough prime, $y = g^a \bmod p$, and a is the secret key. The ciphertext is $(c_1, c_2) = (g^k \bmod p, My^k \bmod p)$, where $M \in Z_p$ is the message. Now, a major component is the computation of $g_{c_1}(a) = c_1^a \bmod p$, since, given c_2, it allows the computation of the plaintext M [12].

As a second example we consider RSA signature generation and decryption. When signing or decrypting one computes $g_b(d) = b^d \bmod n$, where b is the (hashed and processed) text to sign or respectively the ciphertext, d is the secret key and n is the public modulus, i.e., the product of two primes.

Now, Shamir's secret sharing scheme [13] satisfies the property that

$$\text{key} = \sum_{i \in B} (\text{constant}_{i,B})(\text{share}_i) \tag{7}$$

where B is a subset of the set of all participants, called A, and $|B| = t$, the threshold. So, if Shamir's secret sharing scheme is used and g is homomorphic we obtain, using (6) and (7) that

$$\begin{aligned} g_{\text{input}}(\text{key}) &= g_{\text{input}}\left(\sum_{i \in B} (\text{constant}_{i,B})(\text{share}_i)\right) && (8) \\ &= \prod_{i \in B} g_{\text{input}}(\text{constant}_{i,B} \cdot \text{share}_i) \\ &= \prod_{i \in B} (g_{\text{input}}(\text{share}_i))^{\text{constant}_{i,B}}. && (9) \end{aligned}$$

So, in (9) $g_{\text{input}}(\text{share}_i))$ can be evaluated by the shareholder and sent together with his identity (i.e., i) to a reliable combiner.

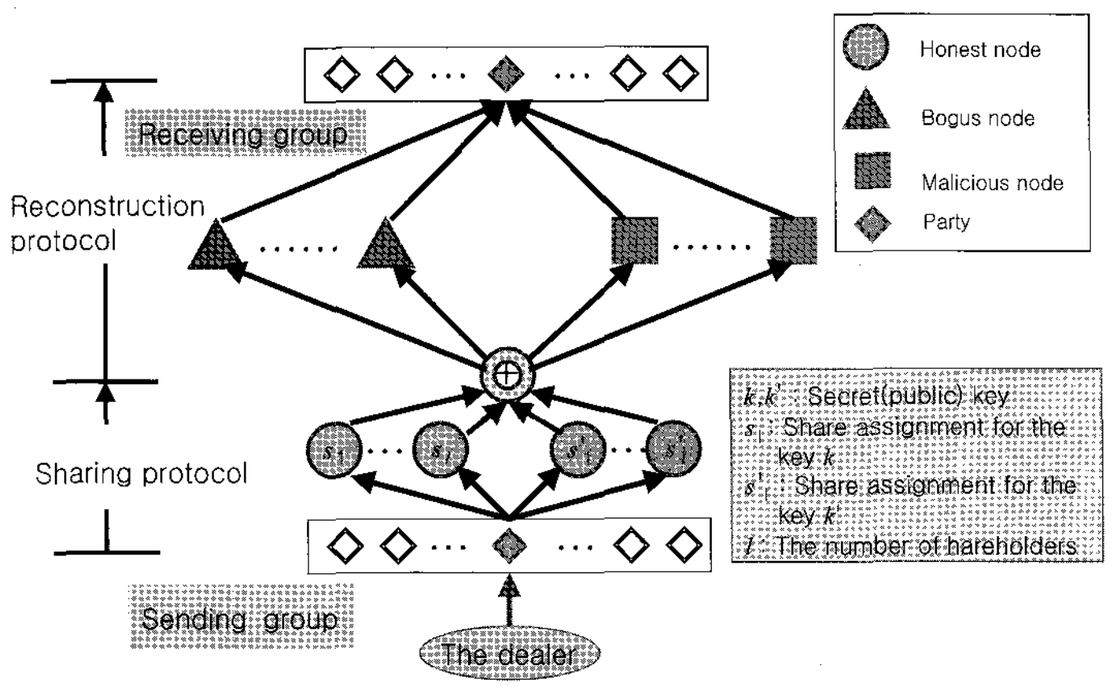

Fig. 7. Byzantine agreement protocol of proactive AVSS.

If enough shareholders responded, the combiner, knowing the identities, can compute a set B' which cardinality must be at least as large as the threshold t, i.e., $|B'| \geq t$. The combiner chooses a $B \subseteq B'$ such that $|B| = t$, computes $\text{constant}_{i,B}$ for all $i \in B$ and then evaluates (9) [7].

E. Byzantine Agreement Protocol with Proactive AVSS

The scheme briefly explains how the use of homomorphic secret sharing is useful towards achieving proactive threshold cryptography.

The scheme assumes that the secret sharing scheme is homomorphic. If $(s_1, s_2, \cdots, s_l$ l: The number of shareholders) is a share assignment for the key k and $(s'_1, s'_2, \cdots, s'_l)$ is a uniformly random share assignment for the "key" 0, then $(s''_1, s''_2, \cdots, s''_l) = (s_1 + s'_1, s_2 + s'_2, \cdots, s_l + s'_l)$ is a new share assignment for the same key k. Assume that one trusts t shareholders. Then, t participants, denoted by j, can each contribute their own random $(s'_{j,1}, s'_{j,2}, \cdots, s'_{j,l})$ When working in an Abelian group and when the secret sharing scheme is perfect, the resulting share $s''_i = s_i + \sum_{j \in B} s'_{j,i}$ will be guaranteed independent of the original share s_i, due to the properties the one-time-pad. Both s_i and s''_i are shares of the same key k.

Proactive AVSS scheme consists of two protocols: Sharing protocol (S) and reconstruction protocol (R). The sharing protocol consists of three stages: First, each party waits to receive its share of the secret from the dealer. Next, the parties jointly try to verify that their shares define a unique secret. Once a party is convinced that a unique secret is defined, it locally computes and outputs a corrected share of the secret using the information gathered in the verification stage. In the reconstruction protocol, each party sends its share to the parties in some predefined set R (the set R is an external parameter with the same role). Next, each party in R waits to receive enough shares to uniquely determine a secret, and outputs the reconstructed secret. In order to deal with possibly erroneous shares (in $\sum_{j \in B} s'_{j,i}$), this work uses a algorithm for error correcting of a new Reed-Solomon code based on Newton's interpolation [14]. Thus, the scheme has no probability of error and zero delay in reconstructing [7].

Sending group's party shares assignment for the secret key k and public key k with l (the number of shareholders), and sharing protocol broadcasts all nodes in the networks. If there are

bogus nodes or malicious nodes in the networks, reconstruction protocol broadcasts all nodes in the networks. Therefore, a coordinator(initiator) of zoning process can communicate securely even if there are the corrupted nodes in MANET.

Sharing Phase

1. Protocol for dealer on input a secret s
   - Randomly choose polynomials $f(x) = a_t x^t + \cdots + a_1 x + s$, s (shared secret value), and $r(x) = r_t x^t + \cdots + r_1 x + r_0$ $r(x)$: Random string for sharing.
   - Compute and hand player P_i the values $\alpha_i \equiv f(i)$ and $\rho_i \equiv r(i)$, for $1 \leq i \leq n$.
   - Computer and broadcast the value $A_i \equiv C(\alpha_i, \rho_i)$, for $1 \leq i \leq n$.
   $C(f(i), r(i))$: Commitment function, $C(x, r)$: Similar to secure hash algorithm (SHA)-1.
2. Player P_i verifies that $A_i = C(\alpha_i, \rho_i)$. If the equation does not hold, then he broadcasts a complaint against the dealer.
3. If player P_i broadcasted a complaint then the dealer broadcasts the values

$$\alpha_i, \rho_i \text{ such that } C(\alpha_i, \rho_i) = A_i. \tag{10}$$

4. If the dealer does not follow some step he is disqualified, otherwise conclude that a secret has been shared.

Reconstruction Phase

1. Each player broadcasts the values α_i, ρ_i.
2. Take $t+1$ broadcasts the values for which $A_i = C(\alpha_i, \rho_i)$ and interpolate polynomials $\bar{f}(x)$ and $\bar{r}(x)$ of degree at most t that pass through those points.
3. Compute $\overline{\alpha}_i = \overline{f}(i)$ and $\overline{\rho}_i = \overline{r}(i)$ and verify that $C(\overline{\alpha}_i, \overline{\rho}_i)$ for all i. If yes, output $\overline{f}(0)$ else output 0.

V. ZONE-BASED SELF-ORGANIZED CLUSTERING ALGORITHM

The proposed algorithm is distributed algorithm which each zone executes the algorithm independently and the failure of one zone does not affect the rest of the network. Zone building with IDs, zone$_i$ (ID$_i$), has cluster G_i with sequential cluster reformations. Cluster G_i has the nodes with one cluster head of unique ID. Zone-based clusters delegates the nodes in each cluster as requirements. Security of zone building with IDs complies with Byzantine agreement protocol.

Therefore, zone-based clusters do not broadcast their informations to nodes in an entire MANET while they need to communicate each other. Securely, they broadcast their informations only to the nodes in zone-based cluster when they are required to communicate, so the interference in a zone with unique ID is less than it of traditional self-organized cluster.

A. Process of the Proposed Algorithm

Zone setup initiator can securely send *setup_init, setup_request, setup_permit, setup_join, setup_accept, setup_report* messages to zone setup participant in station transition of a node. The proposed algorithm executes in five consecutive phases.

Phase 1:

After power-on, listen for beacon signals from cluster heads. If, after a period of time t_b, no signal is received, they begin to transmit beacons themselves in random time slots at a predefined power level. Nodes that receive signals from other nodes in a zone with unique ID measure their distances to the neighboring nodes and calculate the interference factors. If a node receives signals from more than N_u neighboring nodes, only the closest N_u nodes are picked as neighbors. If a node waits for a period of time t_w without detecting a new neighbor, this node assumes that it has detected all of its neighbors in a zone with unique ID and goes to Phase 2. t_w is a parameter that needs to be optimized to reduce waiting time.

Phase 2:

Each node in a zone broadcasts its information, including its interference factor, node ID, zone ID, and a list of neighboring nodes, to other zones in MANET. Each zone also maintains a table that contains information received from other zone. Again, if a zone waits for a period of time T without receiving new information, it assumes the broadcasting has completed and goes on to Phase 3.

Phase 3:

Once broadcasts have completed in a zone, all nodes that can be reached by one or multiple hops will have tables with entries from other nodes that can be reached in a zone. Each node that has a table should perform the following operations. N in a zone is the number of entries in the table.

For $i = 1$ to N

1) Find the node with the minimum interference factor in the table and mark it as a cluster head.
2) Delete the cluster head's neighboring nodes in a zone from the table
3) Stop the iteration when there are no more cluster heads to pick.
4) If the iteration stops in a zone, the size of zone is threshold (t).

It should be noticed that without information losses, each node that can be reached by one or multiple hops should have the same table.

Phase 4:

Nodes that are marked as cluster heads in Phase 3 become the self-claimed cluster heads in a zone and start to send beacons.

Phase 5:

Each remaining node picks the closest cluster head, joins the cluster, and makes a zone. Once every node belongs to at least one cluster, nodes belonged to at least one cluster are deleted due to reforming in zone with unique ID.

VI. SIMULATION RESULTS

The simulations assume that all nodes are uniformly distributed over a 5 m $\times$ 5 m area. Each simulation is run 20 times with different seeds for the random number generator, and the results presented represent the average values on the 20 runs.

The simulation of an efficiency compares interferences of the proposed algorithm with interferences of a traditional clustering algorithm in Fig. 8. In Fig. 8, the x-axis indicates the total number of clusters found and y-axis shows the total number of nodes in the network. The interferences of a traditional clustering algorithm is higher than interferences of the proposed algorithm

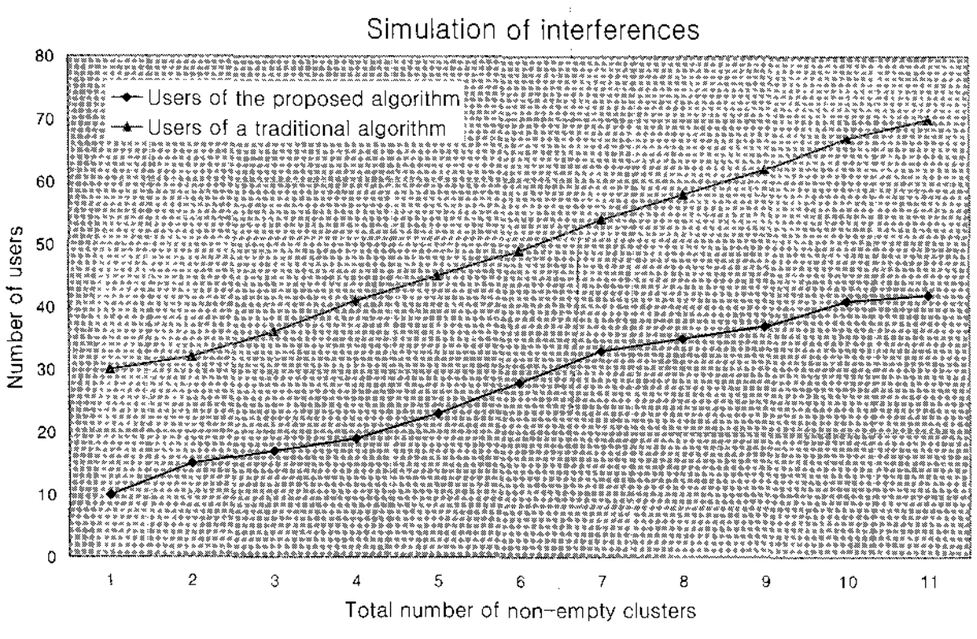

Fig. 8. Comparison of interferences (radio range = 3 m).

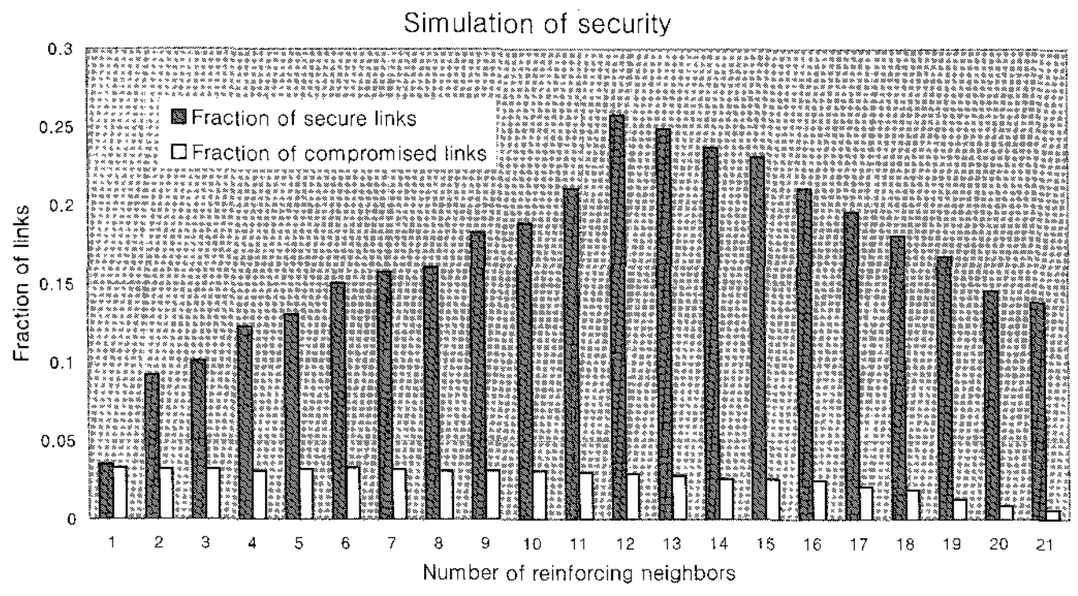

Fig. 9. Comparison of security (radio range = 3 m).

because cluster heads try to pick as many nodes in the radio range as possible in a traditional clustering algorithm.

The simulation of a security compares reinforcement statistics of the proposed clustering algorithm with reinforcement statistics of a traditional clustering algorithm in Fig. 9. In Fig. 9, the x-axis shows the number of reinforcing neighbors and y-axis represents fraction of links in the network. In general, if a link is reinforced by common neighbors, then the adversary must be able to eavesdrop on that link. However, secure links of the proposed Byzantine agreement have more probability than compromised links of a traditional mobile network because secure links of the Byzantine agreement make a decrease in the adversary's probability of compromising a link.

VII. CONCLUSIONS

This paper describes zone-based self-organized clustering that does not broadcast neighbor information to the entire network, on the other hand a traditional self-organized clustering broadcasts neighbor information to the entire network. The zone-based self-organized clustering also considers deleting nodes belonged to at least one cluster and reforming zone with unique ID due to the efficiency (minimizing interferences) of self-organized clustering. Furthermore, this work solves the problem [2] which has not been worked through yet a self-organizing cluster when the cluster head moves out of the cluster in a self-organizing of MANET clustering.

Specially, the Byzantine agreement protocol with proactive AVSS for zone setup initiator of the proposed clustering helps to improve a novel and secure clustering in MANET.

Future work includes developing key managements of secure zone building with ID in MANET.

REFERENCES

[1] M. X. Gong, S. F. Midkiff, and R. M. Buehrer, "A new piconet formation protocol for UWB ad-hoc networks," in *Proc. IEEE Conf. Ultra Wideband Syst. Technol. 2003*, Nov. 2003.

[2] M. X. Gong, S. F. Midkiff, and R. M. Buehrer, "A self-organized cluster algorithm for UWB ad hoc networks," in *Proc. IEEE WCNC 2004*, pp. 1806–1811.

[3] I.-Y. Kim, Y.-S. Kim, and K.-C. Kim, "Zone-based clustering for intrusion detection architecture in ad-hoc networks," Lecture Notes in Computer Science 4238, pp. 253–262, 2006.

[4] S. Sung, "PSS (proactive secret sharing) self-organized clustering scheme with PSU (parallel share updates) in MANET (mobile ad hoc networks)," in *Proc. the 7th APIS 2008*, Jan. 2008, pp. 530–533.

[5] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Programming Languages and Syst.*, vol. 4, no. 2, pp. 382–401, 1982.

[6] D. Dolev, C. Dwork, O. Waatz, and M. Yung, "Perfectly secure message transmission," *J. ACM*, vol. 40, no. 1, pp. 17–47, Jan. 1993.

[7] S. H. Sung, E. B. Kong, "Byzantine agreement with threshold cryptography in unknown networks," in *Proc. SAM 2004*, pp. 68–74.

[8] R. Canetti, "Studies in secure multiparty computation and applications," Ph.D. Dissertation, The Weizmann Institute of Science, June 1995.

[9] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to share a function securely," in *Proc. STOC 1994*, May 23–25, 1994, pp. 522–533.

[10] N. Jacobson, *Basic Algebra I.* W. H. Freeman and Company, New York, 1985.

[11] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, pp. 469–472, 1985.

[12] Y. Desmedt and Y. Frankel. "Threshold cryptosystems." Lecture Notes in Computer Science 435, pp. 307–315. Springer-Verlag, 1990.

[13] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, Nov. 1979.

[14] U. K. Sorger, "A new Reed-Solomon code decoding algorithm based on Newton's interpolation," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, Mar. 1993.

Soonhwa Sung received the Ph.D. degree in 2005 from the Department of Computer Engineering, Chungnam National University, Republic of Korea. From 2000 to 2005, she was teaching in the Department of Computer Web Informations, Daeduk College, Republic of Korea. From 2002 to 2005, she was teaching in the Department of Computer Engineering, Chungnam National University, Republic of Korea. Since 2006, She has been research professor in Chungnam National University, Republic of Korea. Her research interests include information security, ubiquitous computing security, traffic solution of wire and wireless networks, and user authentication system of future internet.