International Journal of Latest Trends in Engineering and Technology
Vol.(10)Issue(1), pp.281-286
DOI: http://dx.doi.org/10.21172/1.101.49
e-ISSN:2278-621X

A DYNAMIC HIERARCHAL FAULT DETECTION APPROACH FOR MOBILE AD HOC NETWORKS

Raman Kumar[1], Parveen Kumar[2]

Abstract: - **Mobile Hosts (MHs) are increasingly becoming common in distributed systems due to their availability, cost, and mobile connectivity but large amount of checkpointed data are not stored on local MHs memory. The limited stable storage available in mobile environments can make traditional fault tolerance techniques unsuitable. Since storage on a mobile host is not considered stable, most protocols designed for these environments save the checkpoints on the base stations. Previous approaches have assumed that the base station always has sufficient space for storing checkpoints. If there is not enough storage available, checkpointing may need to be aborted. An adaptive fault tolerance protocol is described in this paper for manages storage effectively. In cluster based architecture, whole network is divided into several cells and each cell has a Base Station(BS). Cluster has more than one BS it. BSs are the nodes that are given the responsibility for routing the messages within the cell and performing the data aggregation. The communication between two adjacent cells are conducted through the Base Station.**
Keywords: Mobile Adhoc Network, Cluster, Cluster Head

1. INTRODUCTION

A MANET can be organized into a number of clusters in such a way that every node is a member of at least one cluster. A cluster is defined as a group of nodes that are close to each other. The criteria of `close' is that a node in the cluster, the cluster head, has all other members, known as citizens, in its 1-hop vicinity. As a special case, a node that cannot be reached by anyone else forms a single node cluster, or SNC. The size of a cluster is defined as the number of nodes in the cluster and is denoted as SC. It is imperative that cluster head assignment be fair and secure. By fairness, it means that every node should have a fair chance to serve as a cluster head. Note that fairness has two components, fair election, and equal service time. Thus, fair election implies randomness in election decision, while equal service time can be implemented by periodical fair re-election. By security, it means that none of the nodes can manipulate the selection process to increase (or decrease) the chance for it (or another node) to be selected. Obviously, if randomness of the election process can be guaranteed, then security is also guaranteed. For example, the Leader Election (or cluster organization) algorithms in choose either a common evaluation function or node-specific utility functions to compute a score for every node, and the node with maximal score is elected as the clusterhead. They do not guarantee the random selection of clusterheads. They may allow a node to advertise a high score for itself, unless care is taken to make the evaluation process verifiable and with non-repudiation. Several techniques have been used to guarantee the fairness and security of the election process. Firstly, each node i contributes a random value Ri to the input. Then a common selection function is used by all nodes to compute a integer from 0 to SC-1 from a total of SC inputs. The output of the election function must have a uniform distribution in [0, SC -1]. The selection function we use is simply the modular Exclusive OR (or XOR) function, i.e., f(R0;R1;R2; :::;RSC-1) = (LSC1 i=0 Ri) MOD SC. A nice property of XOR is that as long as one input is random (i.e., from a well-behaving" node), the output is random. The random values are fully exchanged within the cluster (clique) and the selection function is computed in a distributed manner, i.e., on each node, to decide the clusterhead. This guarantees that the same clusterhead be computed by all cluster members.

Before we describe our clustering formation protocols, we state a few assumptions about the MANET environment.

- Each node contains a unique and ordered identifier.
- Since clusters can overlap, a node can belong to multiple clusters
- The topology remains static during cluster head computation.
- All links are bidirectional.
- Every node can overhear traffic within its transmission range (this is a common requirement by MANET monitoring schemes.
- Neighbor information is always available. Usually this is implemented by periodically broadcasting HELLO messages and listening to the neighbors' response. Given the assumption, we can obtain the number of neighbors of node i. Let us denote the value to be Ni.

[1] Ph.D Research Scholar, Deptt. of Computer Sci. & Engg., Mewar University, Chittorgarh, Rajesthan, INDIA
[2] Ph.D Research Scholar, Deptt. of Computer Sci. & Engg., Mewar University, Chittorgarh, Rajesthan, INDIA

- A secure, fast and reliable node to node communication infrastructure is available. The infrastructure has to be light-weighted because MANET nodes are often resource-constrained. Recently, efficient protocols for MANET are proposed, such as TESLA, which carries only symmetric cryptographic functions. These protocols make certain assumption that loose synchronized clocks are available.

2. SYSTEM MODEL

In mobile distributed network each cells CLs, has a BS and possibly some Mobile Host (MH)s. BS acts as a local coordinator of transmissions within the cell. Each cell is represented by the ID of its BS. For example, Fig. 2 shows a cell based distributed mobile computing systems and there are four cells CL1, CL2, CL3 and CL4. A MH can communicate with other MHs in different cell or in the same cell only through its own BS. This architecture is characterized by two types of messages – inter-cell messages and intra-cell message. The main aim of approach is to efficiently maintain energy consumption of nodes by involving them in multi-hop communication within a particular cell and by performing data aggregation in order to decrease the number of messages transmitted to MSS. Since, normal nodes only communicate with their BS, which in turn, aggregates the collected information and sends it to its cluster heads and cluster head forwarded it to the MSS. In this scheme, BS failures are more critical than those normal nodes. When a BS fails, re-election of BS is performed within the cluster. Such a recovery scheme is a time and energy consuming process. Therefore, to improve the quality and reliability of mobile distributed networks, a fault tolerant mechanism is needed for such BSs.

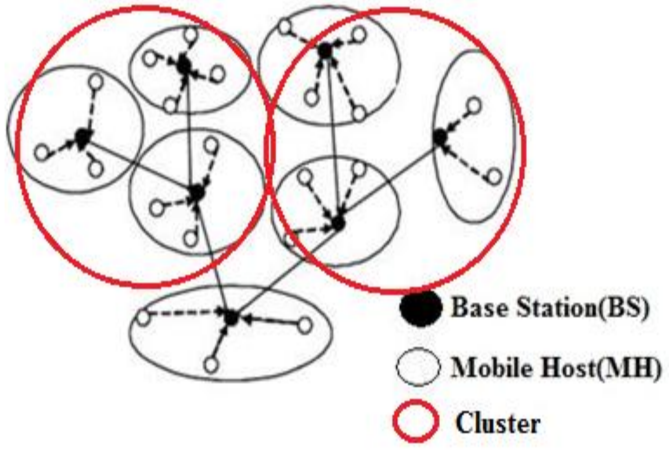

Fig. 2 System Model of Proposed System

We have proposed a non-blocking coordinated fault tolerance algorithm in which MHs take a tentative checkpoint and then on receiving a commit message from the initiator, the MHs convert their tentative checkpoint into permanent. Also whenever a MH is busy, the process takes a checkpoint after completing the current procedure. The proposed algorithm requires fewer control messages and hence fewer number of interrupts. Also, our algorithm requires only minimum number of MHs in a cell to take checkpoints; it makes our algorithm suitable for mobile distributed networks.

3. CLUSTER COMPUTATION PROTOCOL

A cluster is defined as a group of nodes where every pair of members can communicate via a direct wireless link. The cluster requirement can be relaxed right after the clusterhead has been computed. That is, only the clusterhead needs to have direct links with all members. The cluster formation algorithm is used to compute clusters. Once the protocol is finished, every node is aware of its fellow cluster members. We denote the cluster containing i as CL_i, i.e., $\forall j \in CL_i ; CL_j = CL_i$. We define $CL_i^{'} = CL_i - \{i\}$. Once the Cluster Computation Protocol has finished, all nodes enter CLUSTER state.

Nodes periodically request for and receive positions from any neighboring node via the ADV_MSG. On receipt of the ADV_MSG from a neighbor, the node may check for it’s presence in its neighborhood. This protocol processes replies from all nodes. If a node already exists in the neighborhood, its current weight is increased by a scalar, thus lengthening its lifetime. Another technique of reflecting the current topology more accurately allows the node to snoop the network, listening for routing messages, called GEOROUTE_MSG. Every time the node relays or sends a message to its neighbors, it snoops on its neighbors to listen whether they are successful in forwarding the message. On recording a forwarded message, the node rewards the neighbor by increasing the weight. Figure 5 demonstrates this technique. Node A has relayed a message to Node C and is overhearing the network messages. It hears Node C forwarding the message to some node X, and increases the weight of Node C.

The initial weight assigned to a neighbor as well as the subsequent increments may be pre-determined; or, they may be calculated at run-time depending on the density of the network or the load of messages passing through the node. For e.g., if a node is a part of a dense network, some nodes will be in a relatively small region. Thus, a node may age faster and be

removed from the neighborhood. In this case, the initial weight and increments may be small, so that the weight becomes zero as soon as possible. Conversely, in regions having a less dense population, each active neighbor is a precious resource. Hence, care must be taken to ensure that its lifetime in the neighborhood is as long as possible. Similarly, a node involved in relaying a high number of messages is utilizing more energy. It should not be forced to participate in further relaying. Thus, smaller weight and increment values may be assigned to it.
The Neighborhood is a component that is used by the routing component. It operates in two phases; the first phase sets up the neighborhood graph for the node, whereas the second phase updates this graph continually. This ensures that any changes in the topology of the network are detected and eventually reflected in the neighborhood of the node.

```
Neighbor Structure:
struct Neighbor{
unsigned integer NodeID;
Position NodePosition;
unsigned integer NodeWeight;
}
NeighborTable
list of Neighbor;
BuildNeighborhood
Input: NodeID, Position, TimerFrequency, ADV_MSG
Output: NeighborTablePointer
Internal: NeighborTable, InitialWeight, Increment, Decrement
Algorithm:
Neighborhood.Build(NodeID, Position)
{ Set Timer to every TimerFrequency secs;
ADV_MSG.mode := REQ;
ADV_MSG.node := NodeID;
ADV_MSG.position := Position;
return NeighborTablePointer;
}
Neighborhood.Timer.fired()
{ ADV_MSG.destination := ALL;
Neighborhood.Send(ADV_MSG);
forAll Neighbors, n, in NeighborTable
{ n.NodeWeight -= Decrement;
}
}
Neighborhood.Send(ADV_MSG)
{ if(ADV_MSG.position != NULL)
Send ADV_MSG to ADV_MSG.destination;
}
Neighborhood.Receive(ADV_MSG)
{ if(ADV_MSG.mode == REQ)
{ ADV_MSG.mode := REP;
ADV_MSG.destination := ADV_MSG.node;
ADV_MSG.node := NodeID;
ADV_MSG.position := Position;
Neighborhood.Send();
} elseif (ADV_MSG.mode == REP)
        { if (ADV_MSG.node is present in NeighborTable)
        { Get Neighbor, n, in NeighborTable;
        n.NodePosition := ADV_MSG.position;
        n.Nodeweight += Increment;
        }
        elseif (NeighborTable is not full)
        { Neighbor n.NodeID := ADV_MSG.node;
        n.NodePosition := ADV_MSG.position;
        n.NodeWeight := InitialWeight;
        NeighborTable.insert(n);
        }
        }
}
```

The routing component initializes the Neighborhood component at the start. The Neighborhood component initiates the building of the neighborhood using the Neighborhood.Build(…), where it sets a timer to periodically broadcast the advertisement messages. The Neighbor.Receive(…) is an event that is fired whenever a ADV_MSG is received. It processes the message and either updates the Neighbor Table or replies to the ADV_MSG request. Messages are sent using the Neighborhood.Send(…) command.

The Neighborhood Maintenance is performed by snooping over messages in the network. Whenever the routing component transmits a message, it waits for an acknowledgement. On receipt of the acknowledgement, the routing component invokes the Neighhborhood.Maintain(…) command to update the weight of the node to which the message was relayed. It can be seen that neighborhood maintenance is carried out at two levels; the first consists of a reply to periodic requests for node positions, while the second allows the routing component to interact with the neighborhood component

3.1 Cluster head Computation Protocol

The purpose of this protocol is to randomly select one node in the cluster as the cluster head. Without loss of generality, the behavior on the i-th node is discussed as follows:.

1. Generate a random integer R_i.
2. Broadcast a message ELECTION_START=(ID_i, HASH(ID_i,R_i)) to $CL_i^{'}$. HASH is a common hash function. A corresponding timer T_1 is setup.
3. On Receiving all ELECTION_START from $CL_i^{'}$, broadcast the message ELECTION = (ID_i, R_i) to cluster $CL_i^{'}$.
4. If T_1 is timeout, every node for whom ELECTION_START has not been received is excluded from CL_i.
5. On Receiving ELECTION from node j, verify its hash value matches the value in the ELECTION_START message from j. Store R_j locally.
6. If all R_j from $CL_i^{'}$ have arrived, compute H= SEL(R_0, R_1, R_2,…..,R_{Sc-1}) where SEL is the selection function. Determine the cluster head H as the h-th node in the cluster since all IDs are ordered.
7. If H $\neq$ i(i.e., as a citizen), do the following:
   - Send ELECTION_DONE to H.
   - Wait for ELECTION_REPLY from H, then enter DONE state.
8. Otherwise, as a cluster head, H performs following:
   - Setup a timer T_2.
   - On receiving ELECTION_DONE, verify it is from CLi'.
   - If T_2 is timeout, citizens from whom ELECTION_DONE has not been received are excluded from CL_i. Broadcast ELECTION_REPLY to $CL_i^{'}$ and enter DONE state Once the clusterhead is determined, it copies the cluster member list to a citizen list CT_C. The suffix C denotes the current cluster controlled by the clusterhead.

3.2 Cluster Valid Assertion Protocol

All nodes should perform this assertion periodically in DONE state. There are two parts in this protocol.

a) Since the network topology tends to change in an ad hoc network, connections between the elected cluster head and some citizens nodes may be broken from time to time. If a link between a citizen Z and a cluster head H has been broken, Z will check if it is in another cluster. If not, it enters LOST state and activates the Cluster Recovery Protocol. Also, Z is removed from H's citizen list CTC. If there is no more citizens in cluster C, H becomes a citizen if it belongs to another cluster. Otherwise, H enters LOST state and activates the Cluster Recovery Protocol.

b) Even if no membership change has occurred, the cluster head cannot function forever because it is neither fair in terms of service and unsafe in terms of the long time single-point control and monitoring. We enforce a mandatory re-election timeout, Tr. Once the Tr expires, all nodes in the cluster enters the INITIAL state and start a new cluster head setup round. If the cluster property still holds, the Clique Computation step can be skipped.

3.3 Cluster Recovery Protocol

In the case that a citizen loses its connection with previous clusterhead or a clusterhead loses all its citizens, it enters LOST state and initiate Cluster Recovery Protocol to re-discover a new clusterhead. Again, without loss of generality, we discuss the behavior on the i-th node.

- A request message ADD REQUEST=(IDi) is broadcast with a timer T3.
- A clusterhead H receives the request and replies ADD REPLY=(IDH) only after a short delay Td (0.5s in our implementation). The delay is introduced in hope that a connection has been stable for Td can remain to be stable for a fairly long time.
- Node i replies the first ADD REPLY it received, i.e., ADD ACK=(IDi). And enters DONE state. Additional ADD REPLYs are ignored.
- On Receiving ADD ACK, H adds i into its CTC.

- If T3 is timeout and no ADD REPLY is received, there is no active clusterhead nearby. Node i enters INITIAL state to wait for other lost citizens to form new cliques and elect their new clusterheads.

4. FAULT DETECTION ARCHITECTURE FOR CLUSTER BASED MANET

Each node implements a trained, pre-installed IDS (Anomaly & Signature) in a passive state. It will be activated only if the particular node is elected as either cluster-head or backup. The elected cluster head will perform signature detection on all the member nodes along with running anomaly detection only on the backup node. Similarly the backup will be running anomaly detection on all nodes along with signature detection only on the master.

4.1 Signature Detection by Cluster Head

Signature detection requires maintenance of an extensive database of attack signatures, which in the case of ad hoc network would have to be replicated among all the hosts. Every packet in a signature based approach needs to be compared with the attack signature database. This operation requires O(n) time where n is the number of signatures in the database. The signature database would generally have hundreds of attack patterns. Anomaly detection, on the other hand has fewer comparisons, typically less than twenty parameters are used. Thus it can be concluded that signature detection requires greater computational power as compared to anomaly detection. This election algorithm favors a node that has a better computational power and a better battery power as compared to other nodes in the cluster. So it is decided to run signature detection on the cluster head. For a pre-decided window of time, the cluster head will monitor each node for potential attack signatures. This is done in a round robin manner for all nodes in the cluster. The database of signatures does not need frequent updates. An update is needed only when a new attack has been discovered and its signature needs to be added to the database. The probability of update during a particular ad-hoc session is very rare.

4.2 Anomaly Detection by Cluster Backup:

Anomaly detection model is built on a long-term monitoring and classifying of what is a normal or abnormal system behavior. Ad hoc wireless networks are very dynamic in structure, giving rise to apparently random communication patterns, thus making it challenging to build a reliable behavioral model and it is possible that the anomaly detection model will give a lot of false positives. Thus in such a highly dynamic environment, the simplest and the most reliable technique of anomaly detection is threshold based detection. Initial thresholds are set on the preinstalled IDS for local and network parameters which are to be monitored. The required network audit data can be obtained through SNMP (Simple network management protocol) and local data can be obtained using the operating system kernel logs. The thresholds can be modified in joint consensus with all the member nodes of the cluster. A malicious node may go unnoticed if it drops a few packets intermittently. However, if the threshold has been set appropriately, the potential damage caused by such intermittent packet drops will be acceptable and will not significantly affect the MANET. If a node exceeds a small threshold of such allowed "misbehavior" it will be detected and classified as intrusive.

4.3 Detection between Cluster Head (CH) and Cluster Backup (CB):

Cluster head performs signature detection on all nodes including itself. Similarly cluster
backup performs anomaly detection on all nodes including itself. But a compromised cluster head or cluster backup might have its own IDS disabled. So a second degree of reliability and fault tolerance is added to this system by allowing the cluster backup to perform signature detection on cluster head and cluster head to perform anomaly detection on cluster backup. The backup can perform signature detection for a pre-defined window of time but at a higher frequency than the detection performed by master on the member nodes. The master in turn can monitor the parameters of backup on a random basis for detecting anomalies.

4.4 Intrusion response:

The ideal intrusion response for a wireless ad-hoc network is to isolate compromised node from the rest of the network [8]. Fixed networks implement this using the "electronic quarantine" method by updating the firewalls to block the entry of particular compromised node into the network. In a dynamically changing wireless ad-hoc topology, the centralized solution proposed by the electronic quarantine would not be effective, since the implementation of firewalls may not be feasible. In a cooperative IDS architecture for MANETs, one approach suggests "secret isolation" where all other nodes are informed about the malicious node through their 1 hop neighbors, who then delete all the paths to the malicious node from their routing tables, thus secretly isolating the malicious node. It has been proposed to use dirty / counter certificate method, in which the cluster head / backup can isolate a suspected node from the rest of the network by broadcasting a counter certificate for that node. In this approach three kinds of situations comes:

4.5 Sharing of data:

To have a synchronized database of rules (Signature) and parameters (Anomaly) it is proposed to broadcast table updates at the end of election period by master and backup to all member nodes. This is under the assumption that an update to the signature database is needed only when a new attack has been discovered, and the probability of such an update during a particular adhoc session is very rare. Similarly for an anomaly database the update might be just a revision of threshold for a particular parameter which may also be not that often considering that we are using trained pre-installed IDS for each node and even if it happens, it will not be energy consuming.

5. PERFORMANCE ANALYSIS

We compare our algorithm with [7],[9],[10] and[18]. Comparative table 1 as shown below:

Table 1 : - Comparison with the related work

Parameters	[7]	[18]	[9]	[10]	Proposed Algorithm
Non-blocking	Yes	Yes	Yes	Yes	Yes
Minimum Process	No	No	Yes	Yes	Yes
Supports MANET's	Yes	Yes	No	No	Yes
Number of checkpoints	Less	More	Less	Less	Less
No. of control messages	More	More	Less	Less	Less

6. CONCLUSION

In this paper, we have proposed a minimum process and non-blocking fault tolerance scheme for clustering routing protocols. The protocol proposed by us is non-blocking and suitable for mobile environments. It produces a consistent set of checkpoints. The algorithm makes sure that only minimum numbers of nodes in the cluster are required to take checkpoints; it uses very few control messages. Performance analysis shows that our algorithm outperforms the existing related works and is a novel idea in the field. And finally, it reduces the energy consumption and recovery latency when a BS fails.

7. REFERENCES

[1] D.J. Baker and A. Ephremides, "The Architectural Organisation of a Mobile Radio Network via a Distributed algorithm", IEEE Trans. Commun., vol. 29, no. 11, pp 1694-1701, Nov., 1981

[2] D.J. Baker, A. Ephremides and J.A. Flynn "The design and Simulation of a Mobile Radio Network with Distributed Control", IEEE J. sel. Areas Commun.., pp 226-237, 1984

[3] B.Das, R. Sivakumar and V. Bharghavan, "Routing in Ad-hoc networks using a Spine",Proc. Sixth International Conference, 1997.

[4] B.Das, R. Sivakumar and V. Bharghavan, "Routing in Ad-hoc networks using Minimum connected Dominating Sets",Proc. IEEE International Conference, 1997.

[5] M.Gerla, G. Pei, and S.J. Lee, "Wireless Mobile Ad-hoc Network Routing", Proc. IEEE/ACM FOCUS'99, 1999.

[6] Iman, S.; Adnan, A.; Mohamed, E. In-network fault tolerance in networked sensor systems. In Proceedings of the Workshop on DEPENDABILITY ISSUES in WIRELESs Ad Hoc Networks and Sensor Networks, Los Angeles, CA, USA, 26–27 September 2006; pp. 47-54.

[7] S.Monnet, C.Morin, R. Badrinath, " Hybrid fault tolerance for parallel applicatuions in cluster federation", In 4th IEEE/ACM International symposium on cluster computing and the Grid, Chicago, USA, pp 773-782, April, 2004.

[8] B. Gupta, S.Rahimi and R. Ahmad, "A New Roll-Forward Chekpointing /Recovery Mechanism for cluster federation", International Journal of Computer Science and Network Security, Vol. 6, No. 11, Nov., 2006.

[9] G.Cao and M.Singhal, "Mutable checkpoints : A new fault tolerance approach for mobile computing systems", IEEE Transactions on parallel and Distributed Systems, 12(2), 157-172, Feb., 2001

[10] P.Kumar, L.Kumar , R.K. Chauhan and V.K. Gupta, "A Non-intrusive Minimum process Synchronou Fault tolerance Protocol for Mobile Distributed Systems", ICPWC 2005, IEEE International Conference on Personal Wireless Communications, 491-495, New Delhi, Jan., 2005.

[11] Prakash R. and Singhal M., "Low-Cost Fault tolerance and Failure Recovery in Mobile Computing Systems," IEEE Transaction On Parallel and Distributed Systems, vol. 7, no. 10, pp. 1035-1048, October1996.

[12] Cao G. and Singhal M., "On the Impossibility of Min-process Non-blocking Fault tolerance and an Efficient Fault tolerance Algorithm for Mobile Computing Systems," Proceedings of International Conference on Parallel Processing, pp. 37-44, August 1998.

[13] Koo R. and Toueg S., "Fault tolerance and Roll-Back Recovery for Distributed Systems," IEEE Trans. on Software Engineering, vol. 13, no. 1, pp. 23-31, January 1987.

[14] L. Kumar, M. Misra, R.C. Joshi, "Low overhead optimal fault tolerance for mobile distributed systems" Proceedings. 19th IEEE International Conference on Data Engineering, pp 686 – 88, 2003.

[15] Higaki H. and Takizawa M., "Checkpoint-recovery Protocol for Reliable Mobile Systems," Trans. of Information processing Japan, vol. 40, no.1, pp. 236-244, Jan. 1999.

[16] Parveen Kumar, "A Low-Cost Hybrid Coordinated Fault tolerance Protocol for mobile distributed systems", To appear in Mobile Information Systems.