K-Neigh Backbone Tree based Topology Control for Tactical Mobile Ad-Hoc Networks

Sung-Won Lee, Arun Ranjitkar, and Young-Bae Ko
Graduate School of Information and Communication
Ajou University
Suwon, Republic of Korea
{skyline, arun, youngko}@ajou.ac.kr

***Abstract*—Topology control and maintenance of network connectivity are two important issues that need to be addressed in wireless networks in tactical scenarios. Topology control is more prominent with multi-interface multi-channel (MIMC) tactical mobile ad hoc networks (MANETs) due to mobility of wireless nodes. The nodes in tactical mobile ad hoc networks within non-uniform transmission range require proper management, preservation and maintenance in terms of network connectivity.**

In this paper, we propose a K-Neigh Backbone Tree based topology control for MIMC Tactical MANETs that consists of K-Neigh Backbone Tree topology control algorithm, Backbone Tree Association algorithm, and Topology Maintenance algorithm in decentralized management mechanism. The performance and efficiency of the proposed scheme is evaluated with the QualNet simulator.

Keywords-component; Topology Control, Multi-channel, Multi-interface, Tactical Mobile Ad Hoc Network

I. INTRODUCTION

In recent years, study on the different aspects of mobile ad-hoc networks especially in tactical environment that should work in some of the most extreme conditions has created interest among research communities. Although there are many types of topologies and structures of such networks, each type suffers from weaknesses - low signal strength, transmission power consumption, and interference - at the physical layer of communications. One of the prominent issues in tactical mobile ad hoc networks that are highly discussed is the topology control of the network to avoid the network partitioning in the highly mission-critical applications and its maintenance. To date, several protocols have been introduced for the topology control of wireless mobile networks [1~ 4, 15].

The form of network topology is basically dependent upon the channel/interface availability and assignment. Several researches have been carried out to find the optimal method to make efficient use of the available channel/interface [5, 6]. The channel/interface assignment might be done in the scale of short-term or long-term. Short term assignment allows the network to reassign the channel/interface at regular intervals. This is needed as the assigned channel/interface becomes invalid after a given period of time thus making need for the channel/interface to be refreshed after each interval. On the other hand, long term channel /interface assignment is better for the static networks. In such networks, the topology has to remain unchanged throughout the life of network.

Vehicular Ad Hoc Networks (VANETs) and Tactical Mobile Ad-Hoc Networks have a common representation of the tasks. VANET is a communication technology that creates networks between vehicles and/or roadside gateways to allow information exchange providing information on the condition of roads, the status of traffic and/or any breach of traffic regulation to the drivers on the road for their safety and ease of travel. Tactical mobile ad-hoc network provides the network environment for the highly sensitive mission-critical applications. The system needs to communicate with the different types of tactical data messages, such as C2 messages, surveillance messages, and reconnaissance messages.

Topology control (TC) in mobile ad hoc networks has remained a primary area of study even today because of the dynamic nature of its mobility and the need for the maintenance of node-memberships. Such networks need to cope with the frequent changes in its topology, constraints in its bandwidth, and variable link qualities. Therefore, such networks should possess the ability to self-organizing, self-adapting and self-repairing with dynamic topology maintenance. Shanti et al [8] have delineated some guidelines to be followed while designing an ideal topology control protocol for a mobile ad hoc network. This includes that for a resilient topology, the node mobility management must be fully distributed and it should have the ability to generate well-connected topology for most of the network operational time.

Mobile Ad Hoc networks have bandwidth constraints and variable wireless link qualities in sparsely populated rural areas in particular. The proposed scheme is a “K-Neigh Backbone Tree” topology where K neighbors are linked with the construction of the backbone tree. The procedures for the selecting K nodes among its neighbors are chosen with the given algorithm explained in section 3. All the nodes participating in the backbone construction are termed as backbone nodes. The nodes not involved in formation of the backbone nodes are termed as non-backbone nodes and associated with K-Neigh Backbone Tree through a nearest backbone node.

The rest of the paper is organized as follows. We discuss related works in Section 2. In Section 3, we present our K Neigh Backbone Tree base topology control for multi-interface multi-channel tactical MANETs that are accomplished with channel assignment. The performance evaluation is discussed in Section 4. We conclude our paper and discuss the potential future works in Section 5.

II. RELATED WORKS

Many researches are underway to find efficient methods of topology formation in mobile ad hoc networks with highly mobile nodes especially for the mission critical data. Many of the works are also in the field of considering multi-interface and multi-channel in the mobile ad hoc environment. The main researches on topology control (TC) focus on channel assignment which has a direct impact on the network connectivity and interferences [9]. Due to the dynamic nature of the mobile networks, the link needs constant maintenance. This requires periodic break and make of links between two nodes. In order to make the connection, the nodes need to check the available channel and interface because the link may not be formed otherwise. Subramanian et al [10] presents heuristic approach to minimize the interference in the network when the connectivity of a network is fixed and the number of channels available is fewer than needed.

The network structure consists of network manager, cluster manager and managed node. The network manager monitors and controls the overall network while the comprehensive control mechanisms are distributed to its cluster managers. This is quite opposed to traditional network architectures where a centralized network manager supervises directly all nodes. Thus, the dynamic topology changes can be handled by each cluster manager within the local area and architecture is less sensitive to the loss of connectivity between the managers.

Ko et al [11] have researched on self-stabilizing the network in a multi-channel environment and proposed a self-stabilizing distributed channel assignment algorithm to minimizing interference thus improving the network performance in terms of throughput in wireless networks. In this paper, wireless mobile networks will be treated as multi-hop wireless networks in a multi-interface multi-channel environment for simplification. Several related researches are available on the topology control of multi-interface multi-channel multi-hop wireless networks.

In [12], the purpose is to maximize bandwidth aggregation in multi-interface multi-channel ad hoc networks with dynamic channel assignment scheme with the analysis of traffic load per channel. They have two distinct phases: the *exploration phase* and the *convergence phase*. In the *exploration phase*, the neighbor partitioning algorithm assigns the channels based on the network topology from the root node to edge nodes. Each node partitions its neighbors into several groups, assigning one group to each of its interface with the selected channel. In the *convergence phase*, the load-aware channel assignment algorithm collects the traffic load information in local area and assigns the channel having the least degree of interference. This topology control can guarantee better performance in terms of balance of traffic load between channels. Whenever the interference or the traffic loads changes, the algorithm reassigns the channels that have a high overhead.

The authors of [13] present a topology control scheme that considers environments with a variety of obstacles such as buildings, walls etc. Each node scans its neighbors and organizes its neighbors by link quality and creates an ordered list. Periodically the ordered list is exchanged with neighbors. The node selects and creates the links with the neighbors that have no intersection between the ordered lists of neighbors. The number of selected neighbors is defined as the bounded degree property. However, the impact of dynamic link quality is not evaluated and is believed to have high overheads.

In [14], the self-repairing algorithm has been presented for the tree topology. The purpose is to repair the change of tree topology in local areas enabling content-based routing which allows multi-point communication. If the node detects a link failure with its ancestor, the node starts the repairing algorithm. Simultaneously, the subnet including the node is attached to the tree topology through other edge nodes. In this case, the ancestor list of the subnet may be reversed and connected to the ancestor list of the coupled node. The algorithm minimizes the impact of the topology change into only a restricted area. The tree balancing algorithm is not presented in the self-repairing scheme, and thus the tree topology can become unbalanced. This implies that the performance of tree topology will decrease as the level of the balance decreases.

Blough et. al. [15] argued that optimal topology can be created by having every node keep their number of neighbors below a specific value of K. The value K is chosen in such a way that the entire network is connected with high probability. The K-NEIGH topology control produces a symmetrically (asymptotically) connected graph by addressing technical machinery of [16]. They show that setting K to 9 produces the optimal topology with high connectivity. They also argue this value can be minimized to 6 if applications accept weakly connected network topologies. From the energy efficiency point of view, authors of [17] point out that these values of k in the K-NEIGH protocol look too large and interference among neighboring nodes are also high. They propose K^+ Neigh protocol which results in less optimal value of K (either 3 or 4) without network partitioning. Despite of some advantages such as low interference and network connectivity, these algorithms only focus on topology formation with an assumption of no mobility due to the WSN environment.

To the best of our knowledge, there are some issues in the topology control which need to be addressed as follows:

- **Decentralized Topology Maintenance:** If the centralized network management reacts to the overall dynamic topology, this may result in high overhead. Decentralized maintenance takes the benefit of less overhead with reliability.
- **Maximized Channel Diversity:** The well-designed channel allocation algorithms may enhance the network performance.
- **Survivability and Robustness of Topology:** In tactical mobile ad hoc networks, dynamic topology and high interference may prevent reliable communication. Even in unfavorable situations, the network must be able to maintain its survivability and robustness.

III. K NEIGH BACKBONE TREE TOPOLOGY CONTROL

A. System Design

In order to explain our algorithm based on K-neigh topology control, we need to explain some of the assumption

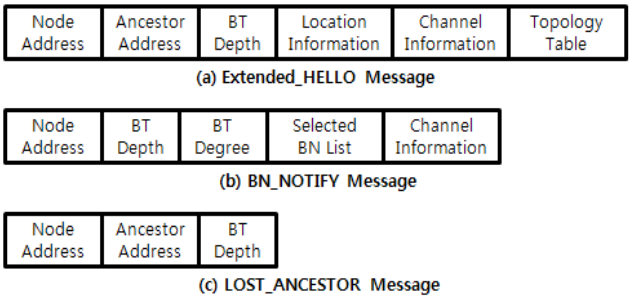


Figure 1. Message Structures

that we had in our system. Each node in the network knows about the two hop neighbor information through the exchange of the extended HELLO messages as explained in following sub-section. All the nodes have knowledge of their location information through either Geographical Positioning System (GPS) or Navigation system attached to them.

K neigh backbone tree topology control algorithm requires knowing the value of K by all the nodes in the network. This K value is recommended as 3 or 4 for the optimum network design as stated in [17]. Another assumption that we have in our system is the number of interfaces of a node and channels the interface can use. All the nodes in our network have three interfaces and can use multiple channels. Among the three interfaces, one interface is fixed with predefined channel which is used by backbone nodes, and every control messages are sent from this interface. The non-backbone nodes will not be using the fixed channel interface except the transmissions of control messages.

The remaining two interfaces are switchable among the remaining channels. First switchable interface is used for the communicating with backbone and non-backbone nodes of the same depth. The second switchable interface is used for communicating with the non-backbone node associated in different depth of backbone tree. According to the depth of backbone node in backbone tree, the channel of first switchable interface is set and a node can switch the channel of second switchable interface over the second switchable interface of the node associated with backbone nodes in different depth and communicate with the node. The K-Neigh Backbone Tree Topology Control algorithm will be described in the coming subsections with the above assumptions.

B. Message Structures

There are three kinds of control messages in the proposed scheme as shown in Fig. 1: *Extended_HELLO*, *BN_NOTIFY* and *LOST_ANCESTOR* messages. The term 'Ancestor' represents the node that only takes part in the backbone tree topology and has the role of the parent node for the non-backbone nodes and/or for the backbone nodes in the lower level. The *Extended_HELLO* message shown in Fig. 1(a) includes its address, ancestor address, depth of the node in the backbone tree (BT DEPTH), the location information, the channel information and its topology table. The topology table maintains neighbor lists specified in the *Extended_HELLO* message. In the initial phase, the Ancestor Address field and channel information for each node is initialized to *NULL* because the backbone tree topology has not been constructed yet.

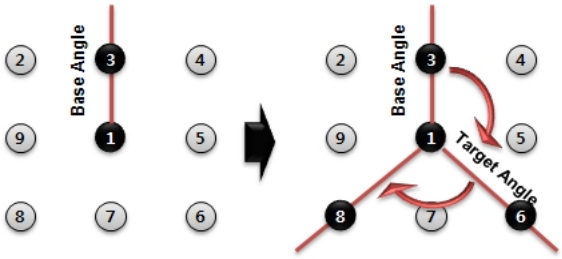
The *BN_NOTIFY* message shown in Fig. 1(b) is used to control backbone tree and only generated by backbone nodes. In network initial phase, only the central node, such as a gateway, can generate and send the message. The *BN_NOTIFY* message includes its address, the information about the current BT depth, a degree of backbone node (BN), the list of backbone nodes selected for next depth of backbone tree and the assigned channel information. If the backbone node needs to create or change the backbone nodes in the lower level, the Selected BN List field is filled with the nodes desiring to join the backbone tree topology. Else, this field is left *NULL*. The Channel Information field has the list of channel per depth in backbone tree which will be assigned in the second interface. The *LOST_ANCESTOR* message shown in Fig. 1(c) includes its address, address of the lost ancestor and the BT depth of the ancestor node. This message is used to inform neighbors about the link failure with the ancestor node. The backbone node related to the ancestor node can recognize the topology change from receipt of *LOST_ANCESTOR* message and perform the maintenance algorithm.

C. K-Neigh Backbone Tree based Topology Control

1) Location-aware Backbone Node Selection Algoirthm

Backbone node selection algorithm considers the K-Neigh based algorithm. In previous research in [15], neighbors are selected up to K value by distance basis, using RSSI estimation. However, this approach can make network partitioning in a certain case, such that the network has partially high node densities in some area by mobility. In this case, each node selects the closest node for neighbor according to the RSSI value and can make the groups with the center of highest node density locally

To solve this problem, our backbone node selection algorithm adapts angular separated selection with location-aware basis as shown in Fig. 2(a). Basically, each node periodically exchanges *Exteneded_HELLO* messages which include the location information from GPS of the sender. With the location information of 1-hop nodes, backbone node selection algorithm decides up to K backbone nodes in next depth of backbone tree which are evenly spaced. In detailed algorithm shown in Fig. 2(b), the backbone node sets the Base Angle by the angle toward its ancestor node and the unit of Angle Variation by 360°/K. Since Node 1 in Fig. 2(a) is the starting backbone node for the backbone selection process such as the coordination node of the unit in tactical mobile ad hoc networks, the base angle is taken at the geographic north. For each Target Angle, the node registers the closest node from the angle as neighbor in the Topology Table. We have K = 3, so the target angle is 120° from the geographic north. Node 1 checks the nodes around 120° among the 1-hop nodes. In this registration phase, the proposed algorithm looks up the nodes within 2-hop range in topology table and if all non backbone nodes in the direction of target angle have its ancestor node, then the algorithm selects no backbone nodes for next depth, because the nodes in that direction are already connected with backbone tree and gotten network connectivity. After all, the node makes up to K neighbors including its Ancestor Node in backbone tree.



(a) Angular Separated Selection

Backbone Node Selection Algorithm

```
Load the Topology Table
Load the Neighbor List

Set Base Angle toward Ancestor Node
Set Angle Variation as 360°/K

While ( N from 0 to K )
    Target Angle = Base Angle + Angle Variation * N

    If (No Neighbor for Target Angle)
      Select the closest node from Target Angle
    End_If
End_While

Update the Topology Table
```

(b) Backbone Node Selection Algorithm

Figure 2. Backbone Node Selection Algorithm

In the given Fig. 2(a), the Node 1 is the only one backbone node in initial step. Since the node can select the backbone node up to given K value, the node selects 3 backbone nodes – Nodes 3, 6, 8 – with Backbone Node Selection algorithm and then the node cannot select more backbone node. At this point the Node 1 will stop the backbone link formation and the next level child backbone node will initiate to form the further backbone links. The process follows the similar process to form the backbone links. The location awareness about nodes within 1-hop range and its utilization make K-Neigh Backbone Tree to get high network connectivity with given K value and with almost minimal number of backbone nodes.

2) K-Neigh Backbone Tree Topology Control

The formation of the K-Neigh Backbone Tree is the initial step in the K-Neigh backbone tree topology control. This tree topology is capable of transmission and receipt of BN_NOTIFY message in the network and is controlled by the K Neigh Backbone Tree topology control algorithm and the Backbone Tree Association algorithm with BN_NOTIFY message. In network initial phase, the message is only generated by central node, Node 1, as shown in Fig. 5 (a) and will be used for the K-Neigh backbone tree formation. The central node has responsibility to manage the network parameters, such as the K value and the channel assignment for each depth which are related to the BT Degree and the Channel Information fields in BN_NOTIFY message respectively, and enclose the parameters in a BN_NOTIFY message which are impossible to be changed by others. As the flow of BN_NOTIFY message, the K-Neigh Backbone Tree topology control consists of two aspects. The one is backbone tree control by backbone node and the other is the association with backbone tree by non backbone nodes.

K Neigh Backbone Tree Topology Control

```
If ( Receipt of BN_NOTIFY message )
  If ( Backbone Node )
    If ( Is included in the Selected BN List )
      Process Backbone Node Selection Algorithm
      Send BN_NOTIFY message
    End_If ( Is included in the Selected BN List )
    Else
      Become Non Backbone Node
      Process BT Association Algorithm
    End_Else
  End_If ( Backbone Node )

  Else If ( Non Backbone Node )
    If ( Is included in the Selected BN List )
      Become Backbone Node
      Process Backbone Node Selection Algorithm
      Send BN_NOTIFY message
    End_If ( Is included in the Selected BN List )
    Else
      Process BT Association Algorithm
    End_Else
  End_Else If ( Non Backbone Node )
End_If ( Receipt of BN_NOTIFY message )
```

Figure 3. K-Neigh Backbone Tree Topology Control

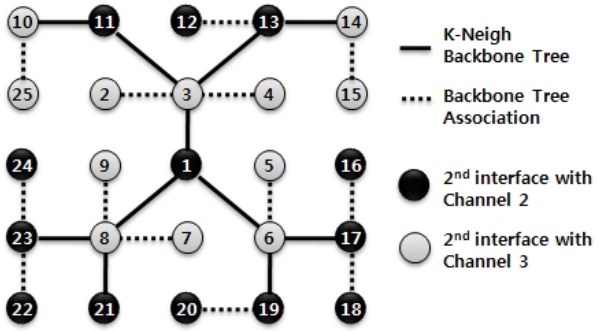
Backbone Tree Association Algorithm

```
Make the list of Backbone Node within 1-hop range

Find Backbone Node(s) with the least node degree
If ( Backbone Nodes more than two )
  Find Backbone Node with the closest distance
End_If ( Backbone Nodes more than two )

Change the channel of 2nd interface
Update the candidate channels of 3rd interface

Update the Topology Table
```

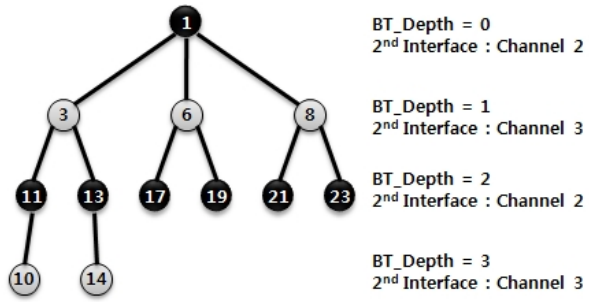
Figure 4. Backbone Tree (BT) Association Algorithm

If the recipient of the BN_NOTIFY message is not a backbone node then recipient node will check if it is included in the selected BN list or not at first as the algorithm shown in Fig. 3. If it is included in the BN list then the recipient will become a backbone node and start processing for the backbone node selection algorithm. The new backbone node will send the BN_NOTIFY message to the neighbors. If the recipient of the BN_NOTIFY message is not included in the selected BN list then it will process the Backbone Tree Association algorithm to make connection with the backbone tree for network connectivity as the algorithm shown in Fig. 4. These processes will be repeated until no more nodes can become the backbone node for the backbone tree formation. This will complete the K-Neigh backbone tree topology control process for the network. The resultant tree topology in K-Neigh Backbone topology is shown in Fig. 5 as the example in grid network environment.

Fig. 5(a) shows the backbone nodes in K Neigh Backbone Tree connected with the solid lines and the non-backbone nodes associated with the backbone nodes with a dotted line. The resultant K-Neigh Backbone Tree is shown in Fig. 5(b) with the depth of the backbone tree. Each of the backbone tree depth have different channel of 2nd interface, which is 1st switchable interface, for the data communication while the control packets are still sent and received through the 1st interface only called by backbone interface.



(a) Grid Deployment Environment



(b) K Neigh Backbone Tree

Figure 5. Example of K Neigh Backbone Tree based Topology (K=3)

To form the backbone tree in the initial phase, the central node, Node 1, decides the network parameters and selects backbone nodes – Node 3, 6, and 8 – in the next depth of backbone tree by Backbone Node Selection algorithm. Including the network parameters and the list of the selected backbone nodes, BN_NOTIFY message is broadcasted in predefined backbone channel that is shared for every node. The recipient, node 3, of the BN_NOTIFY message from node 1 checks whether the node 3 is included in the Selected BN List. If included, the node registers the node 1 as its ancestor node and assigns the channel for 1st switchable interface which is derived from the BT Depth field and the Channel Information field. Then, the Backbone Node Selection algorithm is performed and the node 3 generates and sends BN_NOTIFY message with the list of selected backbone nodes which are nodes 11 and 13. If the recipient, node 4, is not included in the Selected BN List of received BN_NOTIFY message, node 4 performs the Backbone Tree (BT) Association algorithm.

The purpose of the Backbone Tree Association algorithm is to make non backbone node connected with the backbone tree within one hop distance and to make the node density of associating nodes with one backbone node to be distributed, without overhead of control message. The non backbone node lists up the Backbone Node in transmission range and finds the backbone node with the least node degree among them. In Fig. 5(a), node 4, recipient of the BN_NOTIFY message from node 1 but not able to include in the backbone tree will find the backbone node with least node degree, Node 3. If there are more than one candidate nodes as in Node 18, the algorithm selects the closest one from the node. In this case Node 18

Topology Maintenance Algorithm

```
When ( Detection of Link Failure )
  If ( Backbone Node )
    Check the node with Link Failure
    If ( Ancestor Node )
      Send LOST_ANCESTOR message
    Else If ( Backbone Node in next depth )
      Process Backbone Node Selection Algorithm
      Send BN_NOTIFY message
    End_If ( Ancestor Node )
  Else
    Check the node with Link Failure
    If (Ancestor Node )
      Process BT Association Algorithm
    End_If (Ancestor Node )
  End_If ( Backbone Node )

  Update Topology Table
End_When ( Detection of Link Failure )

When ( Receipt of LOST_ANCESTOR message )
  If ( Backbone Node )
    Check the lost node
    If ( Ancestor Node )
      Forward LOST_ANCESTOR message
    Else If ( Backbone Node in next depth)
      Process Backbone Node Selection Algorithm
      Send BN_NOTIFY message
    Else If ( From lower depth )
      Forward LOST_ANCESTOR message
    End_If ( Ancestor Node )
  Else
    Check the lost node
    If ( Ancestor Node )
      Process BT Association Algorithm
    Else ( From lower depth )
      Forward LOST_ANCESTOR message
    End_If ( Ancestor Node )
  End_If ( Backbone Node )

  Update Topology Table
Else_When ( Receipt of LOST_ANCESTOR message )
```

Figure 6. Topology Maintenance Algorithm

chooses Node 17 as a backbone node for the association. The selected node, Node 17, is registered as the ancestor node and the channel of 2nd interface is assigned to the same channel with 2nd interface of the ancestor node. In the last step, the alterations by the results of backbone tree association are reflected in the topology table. After initial formation of backbone tree, the backbone node can become the non backbone node if the backbone node receives BN_NOTIFY message from its ancestor node and is not included in the Selected BN List of BN_NOTIFY message due to topology change. And then the node performs the Backbone Tree (BT) Association algorithm.

3) Topology Maintenance

One of the important sections of the topology control is the topology maintenance. The topology control needs to detect the node failure and to detect this environment. The K-Neigh backbone tree topology control has topology maintenance algorithm as shown in Fig. 6. The topology maintenance algorithm is triggered by the detection of the link failure.

The link failure can be detected when the route failure event in routing protocol is occurred, such as the generation of the Route Error (RERR) message in AODV, or when the HELLO messages from a neighbor are not arrived more than 3 times. If the node detects the link failure, the process checks for whether the node is a backbone node or not. If the node is a backbone node then check whether the node with the failed link is its ancestor node or not. The *LOST_ANCESTOR* message is send if it is. If the node with the failed link is the backbone node in the next depth of backbone tree, then start Backbone Node Selection algorithm to recover the backbone tree, send the *BN_NOTIFY* message to notify the selected backbone nodes and update the topology table. If the node with link failure is not a backbone node then check it for ancestor node. If it is ancestor node then execute the Backbone Tree (BT) Association algorithm to associate the node with the nearest backbone link. Finally update the topology table and the detection of the link failure process is complete.

In the process of the detecting and the responding of the link failure, *LOST_ANCESTOR* message is sent which is received by the nodes in the network. The recipients should process the following algorithm to complete the topology maintenance. If the node receiving this message is a backbone node then check for the lost node. If the lost node is ancestor node then forward this *LOST_ANCESTOR* message in the network. If the lost node is not ancestor node then check whether it is a backbone node in the next depth. If it is then start the process of the Backbone Node Selection algorithm to recover the backbone tree in the network and send the *BN_NOTIRY* message. If the lost node is the node from the lower depth then the node should forward the *LOST_ANCESTOR* message. Update the topology table and the process is completed. If the receiving node is not a backbone node then it checks whether the lost node is its ancestor node or not. If the node is its ancestor node then, the backbone tree (BT) association algorithm should be initiated. This process is required to associate the receiving node to the different backbone node to complete the backbone tree topology. If the lost node of the *LOST_ANCESTOR* message is not the ancestor node then check it for the ancestor node for the lower depth. The *LOST_ANCESTOR* message is forwarded to the upper depth if it is from the lower depth. Update the topology table with all the necessary changes by sending the extended HELLO message. From These algorithms efficiently control the K-Neigh backbone Tree topology.

IV. PERFORMANCE EVALUATION

A. Environement and Parameters

A wide variety of system parameters is used for evaluation in MANET. We consider the application of tactical mobile ad hoc networks as the environment for evaluation. In our simulation environment, the network consists of several vehicles having circular movement in a specific radius within a network area of 38Km×38Km. Some parameters such as node velocity and transmission range are referred to the requirements of the next generation Tactical Information Communication Network (TICN) by Defense Acquisition Program Administration (DAPA), Republic of Korea. Table 1 presents the details of our environment parameters.

TABLE I. SIMULATION PARAMETERS

Environment Parameter	Value
Simulation Time	300 seconds
Area Size of Environment	38Km×38Km
Number of nodes	25
Node Velocity	30m/s
Node transmission range	10Km
Mobility Model	Circular Mobility
The Radius of Circular Mobility	1Km

There are three classes of performance comparison.

- **Pure_*n* Class:** All the nodes have *n* interfaces and have *identical channel assignment*, in which the channel assignment of each node is the same (i.e., NIC-1 assigned with channel-1, NIC-2 with channel-2, and so on).
- **NPCA Class:** The topology is formed by the neighbor partitioning channel assignment algorithm, given in [13].
- **KBT_*k* Class:** The topology is controlled and maintained by the proposed algorithms. *k* represents the node degree.

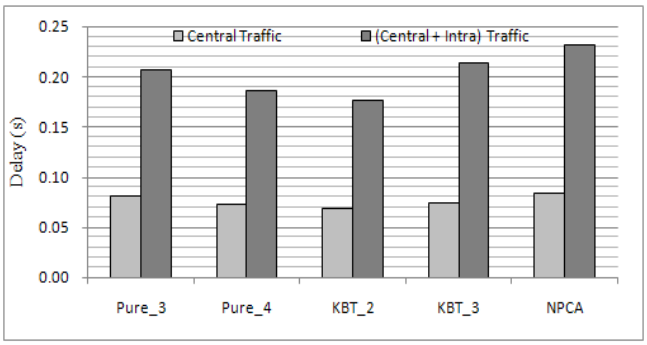
We have also specified two kinds of traffics.

- **Central Traffic:** the Central Traffic includes Command and Control (C2) and Situation Awareness (SA) messages which every node periodically transmits to the coordinating node such as a headquarters of the unit.
- **Intra Traffic:** the Intra Traffic includes the data traffics such as surveillance and reconnaissance which the node generates towards the others.

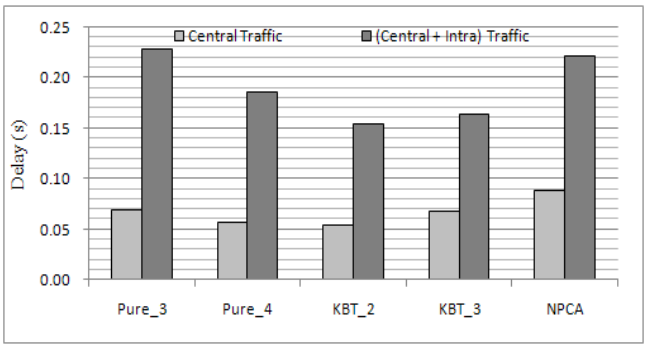
We simulate on QualNet simulator [18] with AODV as our routing protocol. The Central Traffic is configured to 24 which are number of the nodes except central node and the number of the Intra Traffic is configured to 10 randomly. The positions of the coordinating node have been considered as two aspects. In the first, it is in the center of the network whereas in the second, it is at the corner of the network. Each simulation was carried out 5 times for each scenario with a simulation time of 300 seconds. The average results were then considered.

B. Simulation Results

We first present the comparison of the average delay of the successful transmissions in Fig. 7. We simulated our two different traffic classes with central traffic and central with intra traffic in five different classes. In pure class we used two different scenarios with 3 and 4 channels. In KBT class we used 2 and 3 values as K. The last case is the NPCA class. In all the cases KBT_2 gave the least delay with both the traffics. When the coordinating node is at the center as shown in Fig. 7(a) the delay is even lower than that of cases when the coordinating node is in the corner, Fig. 7(b). In all the cases NPCA gave the larger delays because it has dedicated channel and the node degree per interface is limited to 2 or 3. This makes the NPCA class rigid while the KBT_2 class is flexible because the number of channels per interface and the node degree per interface except backbone interface are not fixed.

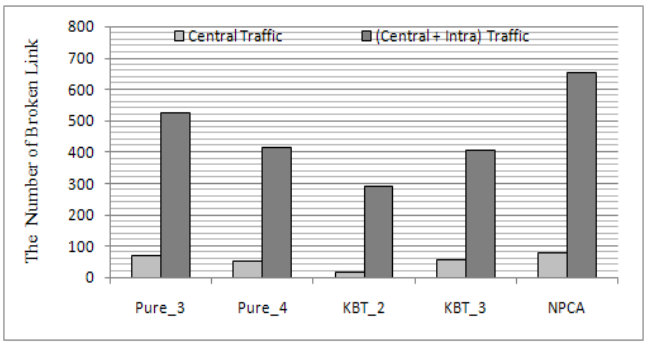


(a) Coordination Node in the Corner

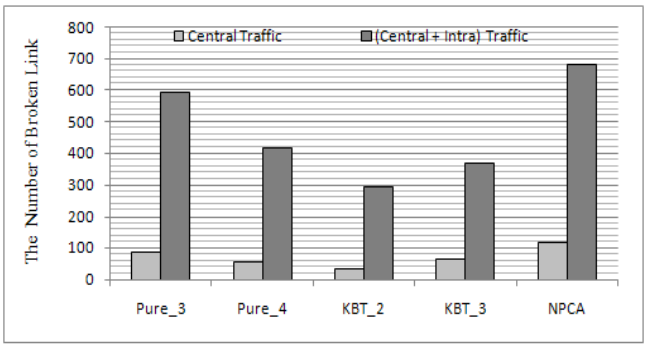


(b) Coordination Node in the Center

Figure 7. Average Delay



(a) Coordination Node in the Corner



(b) Coordination Node in the Center

Figure 8. The number of Broken Links

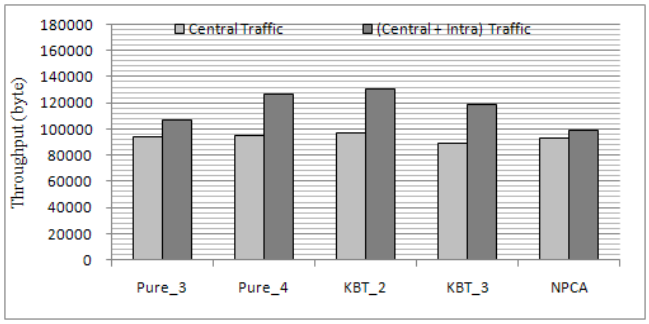
The Pure_*n* Classes show higher delay than KBT_*k* Classes despite a greater number of interfaces. This is because the higher node degree leads to collisions and re-transmissions. Also, the node degree creates queuing delay in each interface without re-queuing the packet with current channel usage.

Fig. 8 shows the number of broken links, defined as the failure of End-to-End connections in TCP due to the generation of RERR message in AODV or the failure of retransmissions of RTS in MAC layer in each class. In scenarios with the coordination node is positioned at the edge of the network as in Fig. 8(a), the link is broken more frequently than in scenarios where the coordination node is positioned at the center of the network, as shown in Fig. 8(b). This is because RTS retransmissions fail in the area near the coordination node due to excessive concentration of data traffic. So, from the receipt of RERR, each node assumes that there is a route failure and restarts the route discovery.

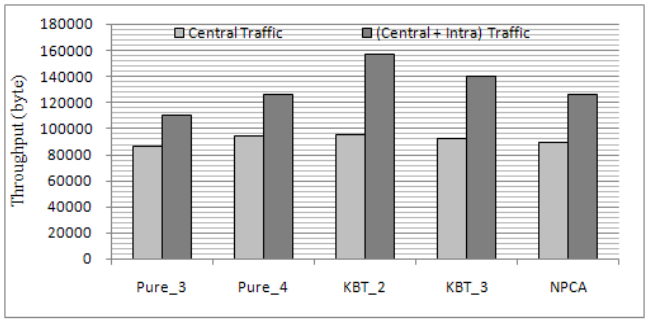
Central traffic with the intra traffic is resulted with the large broken links due to more excessive data surveillance and reconnaissance data. Central traffic only associates with the command and control data with situation aware data. Pure_*n* classes displayed higher broken links due to the higher node degree which creates large broken links. Similarly NPCA is also showing high number of broken links due to the fact that it uses dedicated channel and has the limited links per interface. KBT_*d* has a flexible channel association and robustness.

The overheads of frequent route discovery make a large number of control messages causing the throughput to decrease in the network as a whole as shown in Fig. 9. We defined throughput as the total amount of successful data transmission in the network during simulation time. Among the Classes, the KBT_*d* Classes generally display a higher throughput with less number of broken links than the other classes. This is because the proposed topology control reacts to the topology changes with the decentralized algorithm when the node detects the changes in the topology. The usage of the switchable interfaces and the links with sharing the channel per depth support the redundant links against the broken link due to topology changes. The NPCA Classes show a lower throughput with large number of the broken links. In most cases, they maintain two neighbors per interface or three in some cases. This feature creates weak points when there are topology changes. In case of frequent topology changes, the topology shows unstable connectivity between the nodes.

As shown in Fig. 9(a), the throughput with the coordination node at the corner of the topology gives the higher throughput due to the less frequent link failure compared to the throughput with the coordination node at the center. The throughput in Fig. 9(b) is lower due to the large link failures near the coordination node with excessive concentration of data traffic. The difference between the central traffic and the central traffic with intra traffic is higher in the KBT_*k* than that of the Pure_*n* and NPCA is due to the number of the broken links. If the number of broken links is large then the difference in the throughput is small and vice versa. In KBT_2 the difference number of broken in the two traffics is less than that of the other classes, throughput in this class is higher. Whereas the NPCA shows the least difference in the throughput between the two traffics as it has the largest difference in the link failure count. However, the routing algorithm of AODV in the QualNet simulator does not sufficiently support the advantages.



(a) Coordination Node in the Corner



(b) Coordination Node in the Center

Figure 9. The number of Broken Links

On the other hand, the proposed topology control has higher channel diversity and achieves better performance. Also, the proposed topology control has flexibility in finding a route and shows resilience against topology changes.

V. CONCLUSIONS

We have proposed and evaluated our K-Neigh Backbone Tree based topology control that makes use of a decentralized algorithm for the efficient and robust topology control of tactical mobile ad hoc networks. Instead of making the coordination node solely responsible for controlling the entire topology and deciding the related channel assignment for each node, the Backbone Node Selection algorithm and Backbone Tree Association algorithm are distributed to each local node so that the coordination node only decides the node degree of the network and one-hop topology control. The backbone tree topology constructs the path from each node to the coordination node and ensures the entire network connectivity. The usage of switchable interfaces related to the depth in the backbone tree guarantees redundancy adjustment to instant or permanent topology changes and provides the channel diversity to get higher throughput. The simulation results show the achievement of purposes of our topology control. The routing and forwarding algorithms were out of scope while preparing this paper. It is obvious that the use of proper algorithms for routing and forwarding will allow maximum utilization of network resource. Design of routing algorithms for the maximum utilization of multi-channel multi-interface network resources is left for future work.

ACKNOWLEDGMENT

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2009-C1090-0902-0003)

REFERENCES

[1] M. Bahramgiri, M. Hajiaghayi, V.S. Mirrokni, “Fault-tolerant ad 3-dimensional distributed topology control algorithms in wireless multi-hop networks,” in Proceedings of the IEEE International Conference on Computer Communications and Networks, 2002, pp. 392–397.

[2] S.A. Borbash, E.H. Jennings, “Distributed topology control algorithm for multihop wireless networks,” in Proceedings of the IEEE International Joint Conference on Neural Networks, 2002, pp. 355–360.

[3] Z. Huang, C. Shen, C. Srisathapornphat, C. Jaikaeo, “Topology control for ad hoc networks with directional antennas,” in Proceedings of the A. Nayebi, H. Sarbazi-Azad / Computer Networks 53 (2009) 613–633 631 IEEE International Conference on Computer Communications and Networks, 2002, pp. 16–21.

[4] L. Li, J.H. Halpern, P. Bahl, Y.Wang, R.Wattenhofer, “Analysis of a conebased distributed topology control algorithm for wireless multi-hop networks,” in: Proceedings of the ACM PODC 2001, 2001, pp. 264–273.

[5] A. Hamed Mohsenian Rad and Vincent W.S. Wong, “Distributed Multi-Interface Multi-Channel Random Access,” IEEE Global Telecommunications Conference, 2008, pp 1-6

[6] Xiaoguang Li, Changqiao Xu, “Joint Channel Assignment and Routing in Real Time Wireless Mesh Network,” Wireless Communications and Networking Conference, 2009, pp 1-6.

[7] Young-Bae Ko, Jong-Mu Choi and Nitin H. Vaidya, "MAC protocols using directional antennas in IEEE 802.11 based ad hoc networks," Wireless Communication and Mobile Computing, Apr. 2007.

[8] P. Santi, Topology Control in Wireless Ad Hoc and Sensor Networks, John Wiley and Sons Ltd., 2005.

[9] L. Chen, Q. Zhang, M. Li, andW. Jia, ″Joint Topology Control and Routing in IEEE 802.11-based Multiradio Multichannel Mesh Networks,″IEEE Transactions on Vehicular Technology, vol. 56, Sept. 2007, pp. 3123-3136.

[10] A. P. Subramanian, H. Gupta, and S. R. Das, ″Minimum-interference channel assignment in multi-radio wireless mesh networks,″ Proc. Of IEEE SECON′07, 2007. Pp. 281-490

[11] B.-J. Ko, V. Misra, J. Padhye, and D. Rubenstein, ″Distributed channel assignment in multi-radio 802.11 mesh networks,″ IEEE Wireless Communications and Networking Conference (WCNC), 2007, pp. 3978–3983

[12] Ashish Raniwala, Kartik Gopalan, Tzi-cker Chiueh. "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks", ACM Mobile Computing and Communications Review, Vol. 8, No. 2, 2004

[13] Wattenhofer, Zollinger, "XTC: a practical topology control algorithm for ad-hoc networks", Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, April 2004

[14] Luca Mottola, Gianpaolo Cugola, Gian Pietro Picco, "A Self-Repairing Tree Topology Enabling Content-Based Routing in Mobile Ad Hoc Network", IEEE Transaction on Mobile Computing, Vol 7, No 8, August 2008

[15] D.M. Blough, M. Leoncini, G. Resta, P. Santi, “The k-Neighbors Approach to Interference Bounded and Symmetric Topology Control in Ad Hoc Networks,” IEEE Trans. on Mobile Computing, vol. 5, no. 9, pp. 1267-1282, Sept. 2006.

[16] F. Xue and P.R. Kumar, “The Number of Neighbors Needed for Connectivity of Wireless Networks,” Wireless Networks, vol 10, no. 2, pp. 169-181, 2004.

[17] Dong-Min Son and Young-Bae Ko, "k+ Neigh: An Energy Efficient Topology Control for Wireless Sensor Networks," International Symposium on Systems, Architectures, MOdeling and Simulation Samos (SAMOS VII), Greece, July 16-19, 2007

[18] QualNet Simulator version 4.5, Scalable Network Technologies, www.scalable-networks.com