Design Method Based on Routing Tree for Topology Update in Ad Hoc Network

WEI Zheng-xi, ZHANG Hong, WANG Xiao-ling
Department of Computer Science, Sichuan University of Science & Engineering, Zigong, 643000, Sichuan, China

Abstract

A design method based on the tree-model structure for topology update is presented. The routing tree of every node in network is built by defining the data structure and is used to save the topology information of neighbor nodes. The node topology update is accomplished by exchanging their routing trees. For saving the precious wireless bandwidth, the routing tree is sparsely shaped before sending by pruning the redundant routing information. Then, the node topology update is implemented by using algorithms of inserting and deleting routing sub-trees.

Key words topology update; sparse routing tree; algorithm; ad hoc network

Ad hoc network is a kind of infrastructureless and self-organized mobile network. The wireless communication range of mobile nodes is limited in ad hoc network. The nodes can’t communicate with each other directly, but resort to the other adjacent nodes to forward their packets and exchange information of nodes. Every node not only serves as the mobile terminal, but also is able to store and forward packets[1]. Therefore, the node can be regarded as a router in ad hoc network. It finds the paths for reaching other nodes and maintains the routing information.

As well-known, the topology of ad hoc network changes along with the mobility of nodes. By topology update, one node can find out the connective relation with the other nodes. The qualities of wireless links are much different due to kinds of reasons, so how to update the primary topology is one of the most important problems. There are some effective topology update strategies available in ad hoc network: some based on voluntary routing idea, such as DSDV[2] (Destination-Sequenced Distance Vector), WRP[3] (Wireless Routing Protocol), and so on; and others based on on-demand routing idea, such as DRS[4] (Dynamic Source Routing) and AODV[5] (Ad hoc on Demand Distance Vector). This paper presents a new solution that is based on tree-model data structure for implementing the topology update of mobile nodes.

1 An Overview of Network Links

Fig.1 shows the topology in ad hoc network at one time. Any node in this network can communicate with its adjacent one. If a node wants to communicate with nonadjacent nodes, this node has to rely on other nodes to relay packets. For example[6], there are no direct connective paths between wireless links: B—F, D—E and E—H.

Each node labeled A through H could be considered to be a radio with an associated communication processor. The lines between two nodes indicate the connectivity in the whole network. Suppose the nodes know nothing about neighbor nodes that are more than 1 hop away, they need to exchange connectivity information. The problem is how to exchange topology information and build up a more complete view of the network link’s topology at every node.

2 Design Method for Topology Update

2.1 Routing Tree

Each node should store topology information as a routing tree graph. Fig.2 shows the routing tree for nodes A and C. They are prior to the exchange of any topology information. The routing trees for A and C contain only their nearest neighbors, the nodes which
they can communicate directly. Similar graphs exist for all other nodes. This kind of data structure is called routing tree\(^7\). The data structure of routing tree is defined as follows:

\[
\text{Type RouteTree} = \text{tree}
\]

\[
\text{Tree} = \text{Record}
\]

\[
\begin{align*}
\text{child: RouteTree}; \\
\text{info: elemtp}; \\
\text{next: RouteTree}; \\
\text{parent: RouteTree}
\end{align*}
\]

\[
\text{t=}\text{RouteTree};
\]

As shown above, three pointer fields are created and they are used to connect the other adjacent nodes. However, the “info” field includes a good many sub-options, for example the label of node, the number of links (that indicates the distance between a node and its root node). We could also add any additional options to tree’s info field if need. In this way routing tree’ info is merged into a big field by a number of subfields. The data structure of a routing tree node is shown in Fig.3.

As shown above, three pointer fields are created and they are used to connect the other adjacent nodes. However, the “info” field includes a good many sub-options, for example the label of node, the number of links (that indicates the distance between a node and its root node). We could also add any additional options to tree’s info field if need. In this way routing tree’ info is merged into a big field by a number of subfields. The data structure of a routing tree node is shown in Fig.3.

Fig.3 Logic structure of routing tree

The corresponding relation between routing tree and its logic storage structure is created and shown in Fig.4. This design method builds up a map from a whole routing tree to its structure in storage.

2.2 Sparse Routing Tree

A full routing tree provides enough topology information. Therefore, full topology information can be obtained by exchanging full routing trees. However, the amount of data in the routing tree becomes very large, especially for fully connected networks. For example, for a fully connected network with \(n\) nodes the number of links is \(n(n-1)/2\). Although full routing trees related a node should be stored, exchanging these routing trees may consume too much bandwidth. A smaller copy of the full routing tree (called a sparse routing tree) should be prepared for transmission to neighbor nodes. To reduce the number of branches in the routing tree, the repetitive paths of the nodes in the tree are pruned according to following rules:

1) Only the shortest paths from the root node to another node are retained

2) If there’re redundant paths which are the same number of links from a root node to another node in the routing tree, at most 2 paths are retained.

For example, the link from A to C is same as the link from C to A in Fig.5 (a), so the link from C to A should be removed. Similarly, All redundant identical links can be pruned, as shown in Fig.5 (b).

2.3 Algorithm of Exchanging Routing Tree

To implement topology update, we give two algorithms to exchange information of all nodes in routing tree. The whole procedure is decomposed into two basic operations. The function of \(\text{Insert}(t, x, i, s)\) denotes that the subtree whose root node is ‘s’ is inserted into routing tree whose root node is ‘t’ and located under node ‘x’ of the \(i\)th subtree. Delete\((t, x, i)\) is the reversible operation of \(\text{Insert}(t, x, i, s)\), that is, the subtree under node ‘x’ of the \(i\)th subtree is deleted from the routing tree whose boot node is ‘t’. The description of both algorithms is shown as follows.

Algorithm 1 Insert subtree

Proc Insert\((t, x, i, s)\);
InitQueue\((Q)\);
EnQueue\((Q, t)\);
j:=0;
While not QueueEmpty\((Q)\) Do
\[ t := \text{DlQueue}(Q); \]
\[ p := t; \]
\[ \text{ch\{j\}} := t^.\text{label}; \]
If (ch[j] ≠ x.label) then

While p^.next≠Nil AND not p^.next^.visited
Do

p :=p^.next;
EnQueue( Q, p );
p^.visited :=true;
j:=j+1;

If t^.child≠Nil then
EnQueue( Q, t^.child )

end while
ENDP;

Algorithm 2 Delete subtree
Proc Delete ( t, x, i );
InitQueue(Q);
EnQueue(Q, t ); j:=0;
While not QueueEmpty( Q )  Do

While(p^.next≠Nil) And (not p^.next^.visited)
Do

p :=p^.next;
EnQueue( Q, p );
p^.visited :=true;
j:=j+1;

If t^.child≠Nil then
EnQueue( Q, t^.child )

end while

ENDP;

Fig.5 Topology update for node A

(a) Insert and Merge  (b) Delete and Prune

Nodes in the network gain more topology information by multicasting their individual routing trees to their nearest neighbor nodes[8]. This exchange of routing trees will achieve and filter more complete topology information through the network. For example, assume the routing trees of all nodes initially contain only nearest neighbors, as shown in Fig.2. If node C multicasts its topology information to its all nearest node whose distance is only one hop away from node C, all neighbor nodes would integrate node C’s routing tree into their own. By taking Fig.2 for example, Fig.5(a) shows the routing tree of node A integrated the routing tree of node C. Before saving, any redundant paths in Fig.5(a) should be pruned as shown in Fig.5(b). Simply speaking, the procedure of exchanging routing tree is a combination of operations of Insert and Delete.

3 Conclusions
In this paper, we have discussed the algorithms for exchanging topology information in ad hoc network. Compared with the other ways, it reduces the length of packets by using sparse routing tree and saves the precious bandwidth over wireless link. The design method for topology update has been applied to a tactical Internet project successfully.
References


Brief Introduction to Author(s)

WEI Zheng-xi (魏正曦) was born in Sichuan, China, in 1976. He received the M.S. degree in computer application technology from University of Electronic Science and Technology of China (UESTC). He is currently engaged in the research and teaching with Sichuan University of Science and Engineering. His research interests include computer networks and multimedia technology.

ZHANG Hong (张弘) was born in Sichuan, China, in 1975. He is currently engaged in the management and teaching with Sichuan University of Science and Engineering. His research interests include operating system and computer networks.

WANG Xiao-ling (王晓玲) was born in Sichuan, China, in 1979. She graduated from Sichuan Institute of Light Industry&Chemical Technology in 2002. She is currently engaged in the research and teaching with Sichuan University of Science and Engineering. Her research interests include computer networks.


(Continued from page 105)


 brief introduction to author(s)

ZHANG Jian (张剑) was born in Sichuan, China, in 1977. He received the M.S. Degrees from School of Electronic Engineering, University of Electronic Science and Technology of China (UESTC) in 2003. He is currently pursuing the Ph.D. degree at UESTC. His current research interests include signal processing for communication systems, MIMO, OFDM. E-mail: swordisme@hotmail.com.

HE Zhi-ming (贺知明) is currently an associate professor with School of Electronic Engineering, UESTC. His current research include DSP and radar signal processing.

WANG Xue-gang (汪学刚) is currently a professor with School of Electronic Engineering, UESTC. His current research include radar signal processing and signal detection and estimation.

LO Ka Leong (罗家亮) He was a telecommunication consultant developing software modules for WCDMA network planning tool in Sydney, Australia in 2001. He currently works with UTStarcom, Shenzhen, China. He is an IEEE member. His research interests include MIMO, multi-user detection and TDD-CDMA.