An Improved Time Synchronization Algorithm for Subnet Merge in Ad Hoc Networks

WANG Bo[1], YE Xiaohui[1], ZHAO Yuting[2], GUAN Jianxin[1]

1. College of Electronic Engineering, Naval University of Engineering, Wuhan, China

2. School of Automation, Northwestern Polytechnical University, Xi'an, China

e-mail: wangair1980@yahoo.com.cn, zhaoyuting77@gmail.coms

Abstract: In order to decrease the complexity of popular distributed time synchronization algorithms for subnet merge in ad hoc networks, a new method based on the number and ID of node in subnet was proposed. When two subnets with synchronization difference are close to each other, the method firstly judges the priority of subnets according to the number and ID of node in subnets, and then makes the lower priority subnet synchronize with the higher priority subnet by listening to the time information of the latter. Performance analysis show that the method is better than the popular methods in time overhead and loss of packet because it can ensure the less number of node in lower priority subnet than that in higher priority subnet.

Keywords: ad hoc networks; time synchronization; subnet merge; subnet priority

1 Introduction

Ad hoc networks are temporary networks for instant communication, which have wireless links and work without fixed infrastructure. Due to their decentralized, self-configuring and multi-hopping characteristics, ad hoc networks are suitable for application in these scenarios, including disaster-and-emergency relief, battle field communication, and mobile conferencing and so on.

In ad hoc networks, it is possible that connectivity between two or more sets of terminals loses for an extended period of time. We refer to distinct connected sets of network terminals as subnet. To illustrate, considering an ad hoc network with two subnets: A and B, as depicted in Fig.1. Although terminals in the same subnet are able to maintain consistent time synchronization, time difference between different subnets still exists.

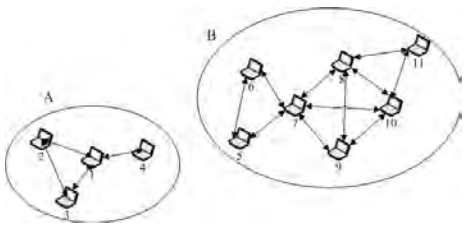

Figure 1. Network partition

Subnets with different synchronization may then come into communications range if they move towards each other. When two terminals in subnets with different time synchronization come into communications range, the message from either terminal may cause collision events in consecutive slots at the other one. With more terminals in different subnets come into communications range, collisions will occur frequently, which makes the capability of communication degrade rapidly.

As depicted in Fig.2, when subnet A and subnet B move towards each other and terminal 4 and terminal 5 come into communication range, the packet transmitted by terminal 5 will be collided at terminal 4 and vice versa.

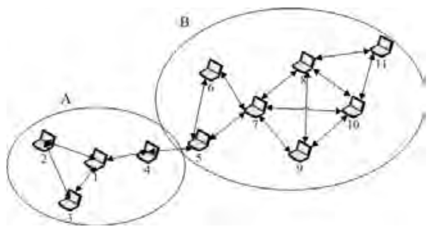

Figure 2. Subnet merge

This problem can be resolved by allowing one of the subnets to adopt the time synchronization of the other subnet.

If all terminals have GPS capability, then the GPS time signal can provide synchronization with error less than 1us. This is sufficient for TDMA communications since the slot durations are typically on the order of 0.1-1ms. However, reliance on GPS for synchronization assumes that all terminals are equipped with GPS receivers, they are in the location where the reception from multiple satellites is possible and the GPS system is operating properly. So it is not suitable for ad hoc networks.

Current clock synchronization methods, which require no external system, for ad hoc networks are divided into master-slave synchronization [1-4] and mutual synchronization [5-8] according to different synchronization principles. The master-slave approach is an open-loop hierarchical approach, in which all nodes align their time to a reference or master node. It has two main limitations. Firstly, the synchronization performance will be seriously influenced if the master node leaves the networks because of mobility or other reasons. Secondly, the synchronization precision of nodes which are away from the master node will be influenced by accumulated errors, especially in large scale networks. Mutual synchronization is a distributed method, in which all nodes try aligning their time to one another, without the need of reference or master node. It can overcome the deficiencies of master-slave approach and is more suitable for wireless ad hoc networks due to its decentralized and distributed nature.

No matter master-slave synchronization methods or mutual synchronization methods, time synchronization can be achieved if and only if the network is connected. However, the network is unconnected for subnet merge, so the methods above-mentioned can not be directly applied to the time synchronization for subnet merge. [9] proposed a time synchronization algorithm for subnet merge, but it is based on multichannel and the nodes only can receive the synchronization message in special channel. The asymmetric partition synchronization protocol (APSP) was proposed in [10]. This method can make one subnet acquire the synchronization of another subnet in an orderly process by utilizing the existing transmission schedule. However, the method assume some form of co-channel interference detection exists in the physical layer, and this can be passed on to higher control layer via special interrupt, which is unable to realize for normal physical protocol.

In this paper, a time synchronization method for subnet merge in TDMA-based ad hoc networks is proposed. This method not only can be used based on single channel, but also can eliminate the deficiency of APSP. So it is simple and applicable.

2 Algorithm Realization

The algorithm is divided into three steps: establishing subnet ID, judging subnet priority and synchronizing lower priority subnet with higher priority subnet. When two subnets get close to each other, the nodes which first receive the message from other subnet are referred to as the trigger nodes (for example node 4 and 5 in Figure 2). Trigger nodes determine the subnet priority according to subnet ID.

For lower priority subnet, the trigger node is responsible for making other nodes in lower priority subnet switch to listening state and wait for a new synchronization message from it. At the same time, the trigger node in lower priority subnet applies to join in the higher priority subnet by listening time information from the later. Then those nodes in lower priority subnet receive new synchronization message from the trigger node and update their time and slot allocation. Thus the synchronization for subnet merge is completed.

2.1 Building Subnet ID

Although the method of building subnet ID in APSP is convenient for judging subnet priority, it ignore the influence of the number of nodes in lower priority subnet on time overhead which is required for synchronization. When the number of nodes in lower priority subnet is larger than those in higher priority subnet, the time overhead required for synchronization must be increased. In order to avoid this, a new method of building subnet ID is proposed.

Table 1.The data needed to be recorded by nodes in subnet

Variable	Function
SubNetId_1	Record the number of nodes in subnet
SubNetId_2	Record the minimum ID of node in subnet
SeqNo	Record the latest serial number of the node with minimum ID
TTL	Ensure SubNetId_2 is used to record the latest minimum ID of node in subnet

This method requires each node in subnet maintain and update the data structure as shown in Tab.1. When the network initializes, the SubNetId_1 records the number of each nodes' 1-hop and 2-hop neighbors, which is known by using neighbor information exchange protocol in USAP, and the SubNetId_2 is the ID of node. When each node receives control message from others, the data will be updated as follows:

SubNetId_1 = (local SubNetId_1 + received SubNetId_1 – common nodes in both);

SubNetId_2 = min (local SubNetId_2，received SubNetId_2);

Finally, by exchanging the information time after time, the SubNetId_1of each node in the subnet equals to the sum of the nodes in the subnet, the SubNetId_2 of each node in the subnet represents the minimum ID of the nodes.

In the initialization process, the nodes in subnet can avoid the redundant information update by comparing the received and local SeqNo. When the node with the minimum ID faults or leaves the subnet, the series number mechanism avoids the expired information continue to transmission. And when TTL=0, the subnet ID needs to be re-established.

2.2 Judging Subnet Priority

In order to illustrate the influence of the number of nodes in lower priority subnet on time overhead for subnet merge, quantitative analysis is carried out.

2.2.1 Time Overhead for Subnet Merge

With no loss of generality, we assume that, as shown in Fig.3, there are N slots in a TDMA frame, and the length of a slot is τ.

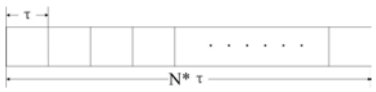

Figure 3. Fame structure of TDMA

We also assume that each node in subnet can only use one slot to send data in a TDMA frame, and the position of the slot follows uniform distribution. So the probability of a node sending data in one frame is:

$$P = 1/N$$

and the average waiting time of the data in a node is:

$$t = \sum_{i=1}^{N} i \cdot \frac{1}{N} \cdot \tau = \frac{(N+1)\cdot\tau}{2}$$

if subnet is h-hop, the average time overhead of synchronization for subnet merge is:

$$T = (h+1)\cdot\frac{(N+1)\cdot\tau}{2}$$

So the time overhead increases with the hop of lower priority subnet.

2.2.2 Judging Subnet Priority

The method of judging subnet priority in this algorithm is depicted as follows: when two subnets close to each other, trigger nodes in different subnets compare their SubNetId_1 by message exchange, and the bigger SubNetId_1 means the higher priority. If two SubNetId_1 are equal, the SubNetId_2 of trigger nodes are compared, and the smaller SubNetId_1 means the higher priority.

The pseudo-code of judging subnet priority as follows:

```
if (local SubNetId_1 > received SubNetId_1)
    local subnet is prior;
else if (local SubNetId_1 < received SubNetId_1)
    neighbor subnet is prior;
else if (local SubNetId_2 > received SubNetId_2)
    local subnet is prior;
else
    neighbor subnet is prior;
end
```

2.3 Synchronization Between Different Subnets

In order to achieve synchronization between different subnets, the trigger node in lower priority subnet must join in the higher priority subnet and synchronize with it, and then, this node sends time information of the higher priority subnet to other nodes in lower priority subnet.

2.3.1 Trigger Node in Lower Priority Subnet Synchronized with Those in Higher Priority Subnet

Firstly, when trigger node find the neighbor subnet is prior, it need to make other nodes in lower priority subnet switch to listening state. This message in MAC layer transmits to the whole lower priority subnet through hop-by-hop.

Then, the trigger node in lower priority subnet acquires the slot allocation schedule of trigger node in higher priority subnet within 2-hop neighbors by listening the control messages of trigger node in higher priority subnet and applies to join in the higher priority subnet by transmitting special message in MAC layer to trigger node in higher priority subnet in an idle slot.

Finally, after receiving the application from trigger node in lower priority subnet, trigger node in higher priority subnet will check whether the application conflict with the slot allocation schedule or not. If conflict-free, trigger node in higher priority subnet reply a confirmed message to trigger node in lower priority subnet. Otherwise, trigger node in lower priority subnet need to apply again.

2.3.2 Synchronizing Lower Priority Subnet with Higher Priority Subnet

After joining the higher priority subnet, trigger node in lower priority subnet transmit “Subnet Synchronization Update” message in MAC layer to its 1-hop neighbors in lower priority subnet by using the idle slot it applied. The subnet ID in header of the data frame is still the lower priority subnet ID, which can avoid the node in higher priority subnet dealing with the message.

Then one-hop neighbors of trigger node in lower priority subnet update their time and slot allocation schedule. Subsequently, each node in neighbors also transmits “subnet synchronization update” message to its 1-hop neighbors. In this way, all nodes in lower priority subnet can synchronize with higher priority subnet step-by-step.

3 Performance Analysis

As shown in Figure 2, if we use the method in this paper, subnet A needs to synchronize with subnet B. On the contrary, subnet B needs to synchronize with subnet A if we use the APSP. So it is obviously that the time overhead of the method in this paper is less than that of APSP. From the other point of view, both the algorithm in this paper and APSP require all nodes in lower priority subnet turn to listening state , which will result in loss of packet in lower priority subnet. The number of nodes in lower priority subnet by using the method in this paper is obviously less than using APSP and so as the loss of packet. Therefore the performance of this method is superior to the APSP.

4. Conclusion

In this paper, a time synchronization method for subnet merge in TDMA-based ad hoc networks is proposed. This method can avoid the problem in APSP and improve the performance by considering the influence of the number of node in lower priority subnet on time overhead required for synchronization and loss of packet. The direction of future research is how to consider the number of node and the hop in lower priority subnet simultaneously when judging the priority of subnets.

5. Acknowledge

I'm greatly indebted to my tutor Professor Ye Xiao-hui who gave me considerable help by means of suggestion, Comments and criticism. Without his pushing me ahead, the composition of this thesis would be impossible. In addition, I deeply appreciate the contribution to this thesis made in various ways by my classmates and friends.

References

[1] Huang L F, Lai T H, On the scalability of IEEE 802.11 ad hoc networks[C], Proc. ACM Mobihoc'02, Lausanne, Switzerland, 2002, P173-182.

[2] Lai T H, Zhou D, Efficient and scalable IEEE 802.11 ad-hoc-mode timing synchronization function[C], Proc. of the 17th International Conference on Advanced Information Networking and Applications, Columbus, OH, USA, 2003, P1-6.

[3] Sheu J P, Chao C M and Sun C W, A clock synchronization algorithm for multihop wireless ad hoc networks[C], Proc. 24th IEEE Int'l Conf. Distributed Computing Systems, Tokyo, Japan, 2004, P574-581.

[4] Zhou D, Lai T H, A scalable and adaptive clock synchronization protocol for IEEE 802.11-based multihop ad hoc networks[C], Proc. Second IEEE Int'I Conf. Mobile Ad Hoc and Sensor Systems, Washington, DC, 2005, P551-558.

[5] Zhu B, Miyano T, Hidekazu M, and Araki K, Study on correlation based timing synchronization algorithm for ad hoc networks[C], The 5th International Conference on Communications and Signal Processing, Bangkok, Thailand, 2005, P593-597.

[6] Tyrrell A, Auer G and Bettstetter C, Fireflies as role models for synchronization in ad hoc networks[C], Bio-Inspired Models of Network, Information and Computing Systems, 2006. 1st, Madonna di Campiglio, 2006, P1-7.

[7] Qi Y and Shi J H, A slot synchronous method and performance analysis of TDMA ad hoc network[C], 2007 IEEE International Workshop on Anti-counterfeiting, Security, Identification, Xiamen, China, 2007, P340-343．

[8] Carlos H R and Thomas K, A mutual network synchronization method for wireless ad hoc and sensor networks[J], IEEE Transactions on Mobile Computing, 2008, 7(5), P633-646.

[9] Snodgrass T E and Stevens J A, Net formation-merging system and method[P], USA：7430192B1，2008-9-30.

[10] Wolf B J, Russell H B and Wang K C, Synchronizing transmission schedules of partitioned ad hoc networks[C], IEEE MILCOM 2007, Orlando, FL, USA, 2007, P1-7.