PATM: Priority-Based Adaptive Topology Management for Efficient Routing in Ad Hoc Networks

Haixia Tan, Weilin Zeng, and Lichun Bao

Donald Bren School of Information and Computer Sciences,
University of California, Irvine, Irvine, CA 92697
{htan, wzeng, lbao}@ics.uci.edu

Abstract. We propose a distributed and adaptive topology management algorithm, called PATM (Priority-based Adaptive Topology Management), that constructs and maintains a connected backbone topology based on a minimal dominating set of the network. PATM provides a succinct presentation of the network topology to routing protocols, and therefore reduces the control overhead in routing updates. Two optimizations are proposed to further reduce the topological information exchanges among the nodes by piggybacking topology updates in packets transmitted by each node, and adaptively adjusting the topology update intervals. The efficiency of the algorithm is validated by simulations based on DSR (Dynamic Source Routing) protocol. The simulation results demonstrate that PATM not only significantly reduces the routing control overhead, but also substantially improves the network data forwarding performance.

1 Introduction

Different from most cellular networks which are supported by a fixed, wired infrastructure, and scheduled by the central base stations, ad hoc networks are self-organizing, self-configuring wireless networks. Topology management has been proposed as an effective and efficient approach to performing some control functionalities in ad hoc networks. The main task of topology management is to select an appropriate subset of the original topological network graph. The backbone constructions are usually based on hierarchical clustering, which consists of selecting a set of clusterheads that covers every other node, and are connected with each other by means of gateways.

Different clustering algorithms propose different criteria in selecting clusterheads and gateways [3][7][11][6][8]. SPAN [12] adaptively elects coordinators according to the remaining energy and the number of pairs of neighbors a node can connect. GAF [1] subdivides a sensor network into small grids, such that only one node in each grid is active at each point of time. PILOT [2] proposed to use a set of mobile nodes in the sensor network to bridge failing connections. In ASCENT [5], a node decides to join the backbone based on the number of neighbors and the data message loss probability at the node. STEM [4] also saves power by turning off a node's radio. It adaptively puts nodes to sleep and to wake up nodes only when they need to forward data.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 485 – 492, 2005.
© Springer-Verlag Berlin Heidelberg 2005

TMPO [6] proposes to construct and maintain a network backbone based on MDS (*Minimal Dominating Set*) and CDS (*Connected Dominating Set*) using only two-hop neighbor information. CEC [7] is another distributed, proactive clustering algorithm. In CEC, Clusterhead election is based on the lifetime of each node. The gateways are elected according to the node's degree. CEC is an improvement of Geographic Adaptive Fidelity (GAF [1]), which relies on location information.

Unlike TMPO, CEC, and WCA [8], On-Demand Cluster Formation (ODCF [11]) is a reactive, on-demand clustering algorithm. Adaptive Clustering (AC [3]) proposed to use clustering for different tasks, such as spatial reuse of bandwidth, Quality of Service (QoS) provisioning by resource allocation within clusters.

We propose a novel hierarchical clustering algorithm, Priority-based Adaptive Topology Management (PATM), which is adaptive to the dynamic changes of topology, bandwidth resource availability and traffic loads. We show that cluster-based control mechanisms can significantly reduce the overhead in routing, while improving data forwarding services. In comparison with other clustering algorithms (such as GAF [1], PILOT [2], and TMPO [6], etc.), PATM distinguishes itself by combining these features: (1) It does not require node position information or synchronization among nodes. (2) It does not need centralized control over the ad hoc network. Every node makes decisions based on its local information. (3) It proactively maintains a connected backbone, but without exchanging control messages periodically. Furthermore, topology updates are dramatically reduced using two optimizations. The first one is to piggyback the small control messages to the ongoing traffic. The second is to adapt the topology update intervals based on the network mobility.

The rest of the paper is organized as follows. Section 2 describes PATM algorithm. Section 3 presents extensive simulation results by running DSR with and without topology management using PATM. Section 4 summarizes this paper.

2 PATM

2.1 Priority Computation

PATM is a distributed clustering algorithm for constructing the connected dominating set of the network by comparing the priorities of two-hop neighbors. The priority of a node, say i, is a function of the node's ID, current time slot number t, the remaining energy Ei and the moving speed of the node Si using the following formula:

$$Pi = h(i, t, Ei, Si)$$

which gives low priority at high speed or low energy situations.

2.2 Information Exchange

PATM requires nodes in an ad hoc network directly exchange priority information of themselves and that of their one-hop neighbors. This change in the information exchange allows the nodes to adapt the interval of their priority computations according to the network traffic and mobility conditions, instead of carrying out the priority computation by other nodes periodically. When network traffic load or mobility varies at different parts of the network, nodes can be more active or passive in forming

the backbone of the network. For example, when a region of the network carries very light traffic, PATM can increase the interval of priority updates, causing less control overhead, and more energy savings.

In addition to exchanging the priority information of the nodes, the clusterhead status of a node and its one-hop neighbors are also exchanged by broadcasts.

i	*Pi*	*i.type*
j	*Pj*	*j.type*
k	*Pk*	*k.type*

Fig. 1. Information in a PATM Update Packet

i	*Pi*	*i.type*
j	*Pj*	*j.type*
k	*Pk*	*k.type*
N_j^1	-	-
N_k^1	-	-

Fig. 2. Node *i*'s Neighbor Table

As an example where node *i* has two one-hop neighbors *j* and *k*, that is, $N_i^1 = \{j, k\}$, node *i* broadcasts a packet with the information shown in Fig. 1. Similarly, node *j* broadcasts the information about itself and N_j^1. So does node *k*.

According to the neighbor information exchanged, every node acquires and maintains a neighbor table that stores the information about its one-hop and two-hop neighbors including the priority and the type of each node. Following the same example given in Fig. 1, the content of node *i*'s neighbor table is shown in Fig. 2.

The last two rows in Fig. 3 is an abbreviation of all the one-hop neighbors of node *j* and *k*, which is dependent on the concrete topology of the network. The corresponding attributes of the members in the last two rows are omitted.

2.3 Clusterhead Election

Without loss of generality, we describe PATM clusterhead election algorithm from node *i*'s point of view. First, node *i* initializes its own type as *host*. Then it decides to become a *clusterhead* if either one of the following criteria are satisfied.

(1) Node *i* has the highest priority in its one-hop neighborhood.
(2) Node *i* has the highest priority in the one-hop neighborhood of one of its one-hop neighbors.

2.4 Doorway and Gateway Election

After the MDS is formed, the CDS is constructed in two steps.

(1) If two clusterheads in the MDS are separated by three hops and there are no other clusterheads between them, a node with the highest priority on the shortest paths between them is elected as a *doorway*, and becomes a member of the CDS.
(2) If two clusterheads or one clusterhead and one doorway are only two hops away, and there are no other clusterheads between them, one of the nodes between them

with the highest priority becomes a *gateway* connecting the clusterhead to another clusterhead or the doorway to the clusterhead, and becomes a member of the CDS.

As an example, Fig. 3 (a) shows the topology of an ad hoc network. Fig. 3 (b) shows a possible result of applying topology management and forming the CDS.

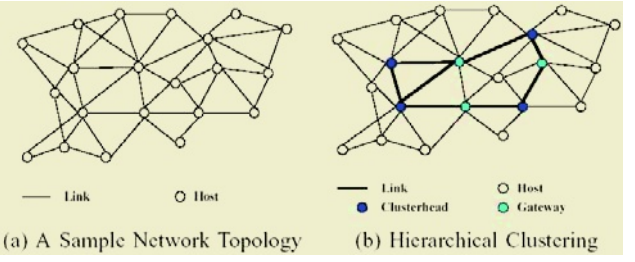

Fig. 3. Topology Management

2.5 Piggybacking Optimization

In PATM, nodes in networks have to exchange routing control information to maintain the connectivity of the networks in mobile environments. Therefore, instead of sending out topology management update packets alone, we apply the piggybacking mechanism to the ongoing topology update packets whenever necessary. The outgoing packets are those sent by the network layer, which could be regular data packets, or routing control messages, such as Route Request (RREQ) and Route Reply (RREP) in DSR.

2.6 Adaptation to Network Mobility

The adaptation of PATM to network mobility is based on a key observation that the interval between re-computing the node priorities and sending topology updates is critical for the network performance. If the interval is too short, the control packet overhead will increase dramatically, and if the interval is too large, the CDS in PATM may not be able to catch up with the topology change. In PATM, the interval of re-computing the node priorities and updating neighbor information varies at different nodes. Each node determines its own interval value.

The frequency of one-hop neighbor changes during the current update interval is taken as an indicator of the relative speed in deciding the next update interval in PATM. As shown before, every node maintains a neighbor table in PATM. The number of one-hop neighbor changes during the current interval *Ti* is used to update the next interval value *Ti* for topology updates. In addition, if node *i* has not received any packets from a one-hop neighbor for a certain period of time, this one-hop neighbor and its associated two-hop neighbors will be deleted from the table.

Fig. 4-6 describe the essential functions of PATM using C-style pseudo-codes. Fig. 4 provides the initialization of various variables in PATM. Fig. 5 specifies the callback function after each update interval to adjust the next update interval. For conven-

ience, the factors adjusting the interval value are given in the algorithm, which performs well in the simulations. However, they are tunable parameters. Fig. 6 provides the condition for piggybacking topology updates to the outgoing packets.

```
Init(i) {
1  piggybacked = FALSE;
2  oneHopChangeNum = 0;
3  Ti = 10;
4  Schedule(Callback, Oi);
}
```

Fig. 4. Initialization in PATM

```
Callback(i) {
    // Re-compute priority
1   t = Current_time();
2   Pi = h(i, t, Ei, Si);
3   if (!piggybacked)
4     Propagate_topology();
5   piggybacked = FALSE;
6   Check_one_hop_nbr();
7   if(oneHopChangeNum <= 1)
8     Ti *= 2;
9   else if (oneHopChangeNum <= 4)
10    Ti *= 1.5;
11  else if (oneHopChangeNum >= 8
      && oneHopChangeNum <= 15)
12    Ti *= 0.75;
13  else if (oneHopChangeNum > 15)
14    Ti = 30; //Ti is 30 seconds.
15  oneHopChangeNum = 0;
    // Schedule the next callback.
16  Schedule(Callback, Ti);
}
```

Fig. 5. PATM Function for Maintenance

```
Piggyback(i) {
1  if (Current_time()-piggyback_time
      > Ti-2) {
2    Piggyback_topology( );
3    piggyback_time = Current_time();
4    piggybacked = TRUE;
5  }
}
```

Fig. 6. PATM Function for Piggyback

Before any outgoing packet is sent down to the network interface, function `Piggyback()` is invoked to see if there is a topology update packet ready to piggyback. The variable `piggyback_time` records the time when a piggyback happens (`Piggyback()` line 3). To prevent the piggyback procedure from happening too frequently, piggybacking happens only if the time difference between the current time and `piggyback_time` is greater than a threshold (`Piggyback` line 1). In addition, the same topology information does not have to be piggybacked in every outgoing packet because the transmission delay increases as the size of each packet increases (`Piggyback`() line 4 and `Callback`() line 3-4).

If the topology information has not gotten an opportunity to be piggybacked during the period, node *i* will broadcast it in a separate control packet (`Callback` line 3-5) so as to guarantee the topology information is broadcast at least once in a period.

In Fig. 5, function `Check_one_hop_nbr()` is not specified, but is used to check the validation of every element in N_i^1. Variable `oneHopChangeNum` is used to record the number of one-hop neighborhood changes during the current update period. The variable is set to 0 in the initialization (`Init()` line 3) and at the beginning of each period (`Callback` line 15). Thereafter, every time when a one-hop neighbor is inserted or deleted from the neighbor table, the variable will be increased by 1. At the end of the period, the value of `oneHopChangeNum` is checked as the relative speed to adjust the length of the next period (`Callback` line 7-14).

3 Performance Evaluation

3.1 Simulation Environment

We simulate PATM by combining with the Dynamic Source Routing protocol (DSR), which is a reactive unicast ad hoc routing protocol, using NS-2 simulator [10]. The major control overhead of DSR is caused by Route Request packets (RREQs) which are flooded in the network in search of paths to the destinations. Therefore, we modify the Route Request phase such that every node rebroadcasts an RREQ packet if the node is not a host. As a result, hosts are excluded from intermediate nodes for a routing path.

We compare the performance of DSR with three modified DSR versions. Table 1 summarizes the different characteristics of the four routing protocols. We also compare the performance of PATM with another clustering algorithm, SPAN [12]. Both protocols run over DSR.

Table 1. Characteristics of Protocols

Protocol	With piggyback?	Topology adaptive?
DSR	No	No
DSR-PATM-1	No	No
DSR-PATM-2	Yes	No
DSR-PATM-3	Yes	Yes

We use the following metrics to show the performance of each protocol.

(1) *Normalized Control Overhead*: the total number of control packets divided by the total number of data packets delivered to destinations.
(2) *Delivery Ratio*: the total number of data packets delivered to destinations divided by the total number of data packets sent from sources.
(3) *Average Delay*: the average delay of all the data packets delivered to destinations.
(4) *Goodput*: the total number of data packets delivered to destinations during a simulation divided by the time span of the simulation.

3.2 Simulation Results

First, scenarios with different offered load are simulated where the number of CBR sessions varies from 15 to 60. The maximum speed of the nodes is 20m/s. In DSR-PATM-1 and DSR-PATM-2, the value of the interval T is set to 20s for all the nodes.

Fig. 7 shows the performance comparison under different metrics between the four protocols. It is apparent that the PATM-3 with piggybacking and adaptive update interval adjustment improves the delivery ratio, and reduces the routing overhead and average delay in most of the cases.

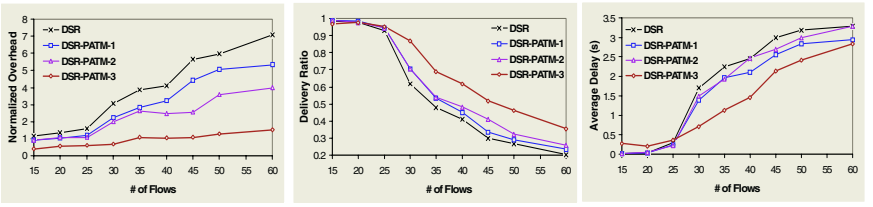

Fig. 7. Performance under Various Loads

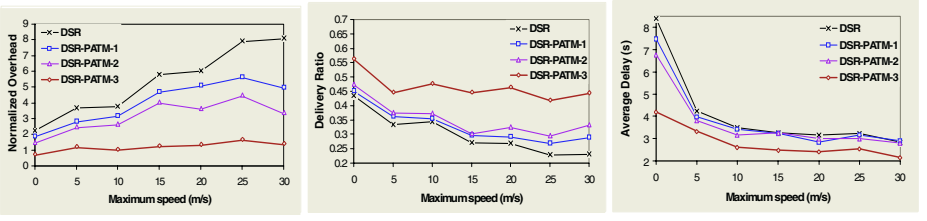

Fig. 8. Performance under Various Speeds

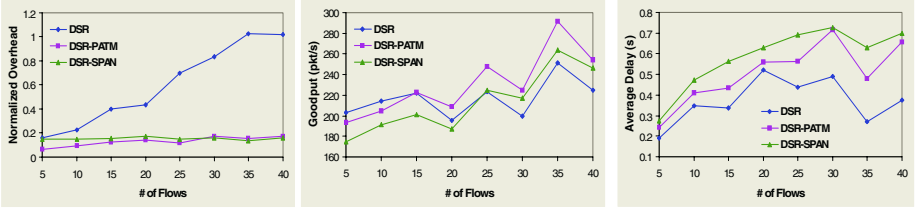

Fig. 9. Performance Comparison between PATM and SPAN

In the second set of simulations, we fix the number of CBR sessions to 50 and vary the maximum speed of nodes from 0 to 30m/s. Fig. 8 show the performance of the four protocols under different speeds. DSR-PATM-3 always performs the best among all the protocols, in both low mobility and high mobility scenarios.

Third, we compare PATM with SPAN. Here TCP flows are used to simulate data traffic. Each TCP flow lasts 900 seconds, which is the length of the whole simulation. Scenarios with different offered load are simulated where the number of TCP flows varies from 5 to 40. The maximum speed of the nodes is 20m/s. Fig. 10 shows the performance comparison, and PATM achieves smaller overhead and delay, and higher goodput.

4 Conclusions

We have presented PATM, a highly efficient topology management approach based on dynamic node priorities and network mobility. PATM builds a backbone of the original network for sufficient network connectivity and efficient data communica-

tion. We have applied it to routing protocols and have shown that it can reduce the control overhead significantly while improving the routing performance. Several optimizations have been applied in PATM such as update piggybacking, mobility-adaptive priority re-computation and topology information update. We show the application of PATM to the on-demand routing protocol DSR. Simulation studies demonstrate that PATM can reduce the routing overhead dramatically while improving the routing performance, in a variety of mobility scenarios with different traffic load.

References

1. Y. Xu, J. Heidemann, D. Estrin. Geography-informed energy conservation for ad hoc routing. Proc. of MobiCom 2001, Rome, Italy, pp. 70-84, July 2001.
2. T. Srinidhi, G. Sridhar, V. Sridhar, Topology management in ad hoc mobile wireless networks. Real-Time Systems Symposium, Work-in-Progress Session, Cancun, Mexico, December 3rd, 2003.
3. C.R. Lin, M. Gerla. Adaptive clustering for mobile wireless networks. IEEE Journal on Selected Areas in Communications, vol. 15, no. 7, pages 1265-1275, 1997.
4. C. Schurgers, V. Tsiatsis, S. Ganeriwal, M. Srivastava. Topology management for sensor networks: exploiting latency and density. Proc. of the 3rd ACM MobiHoc, Lausanne, Switzerland, June 9-11, 2002.
5. A. Cerpa, D. Estrin. ASCENT: Adaptive self-configuring sensor networks topologies. Proc. of IEEE INFOCOM, Jun. 2002.
6. L. Bao, J.J. Garcia-Luna_Aceves. Topology management in ad hoc networks. Proc. of the 4th ACM MobiHoc, Annapolis, Maryland, June 1-3, 2003.
7. Y. Xu, S. Bien. Topology control protocols to conserve energy in wireless ad hoc networks. Submitted for review to IEEE Transactions on Mobile Computing, January 2003. CENS Technical Report 0006.
8. M. Chatterjee, S.K. Das, D. Turgut. WCA: a weighted clustering algorithm for mobile ad hoc networks. ClusterComputing 5, pp. 193--204, 2002.
9. A. Amis, R. Prakash, T. Vuong, D.T. Huynh. MaxMin D-Cluster Formation in Wireless Ad Hoc Networks. Proc. of IEEE INFOCOM, March 2000.
10. NS notes and documentation. http://www.isi.edu/nsnam/ns.
11. Y. Yi, M. Gerla, T.J. Kwon. Efficient flooding in ad hoc networks using on-demand (passive) cluster formation. Proc. of the 3rd ACM MobiHoc, Lausanne, Switzerland, June 9-11, 2002.
12. B. Chen, K. Jamieson, H. Balakrishnan, Robert Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. Wireless Networks: 8 (5): 481-494, September 2002.