A New Route Discovery Algorithm for MANETs with Chase Packets

Mznah A. Al-Rodhaan, Lewis Mackenzie, Mohamed Ould-Khaoua
Department of Computing Science
University of Glasgow
Glasgow, UK G128QQ
Email: {rodhaan, lewis, mohamed}@dcs.gla.ac.uk

Abstract: *We introduce a new route discovery algorithm for MANETs using chase packets. The algorithm works by including the most likely destinations for a given source node in a local neighbourhood and broadcasting route requests at full speed within this region. Outside the neighbourhood however, propagation of the route requests is deliberately delayed to provide chase packets with an opportunity to catch up and minimise network congestion. The algorithm is adaptive and continuously updates the boundary of each source node's neighbourhood to optimise performance. Here, we provide a detailed performance evaluation and compare our algorithm with existing alternatives, to demonstrate that it does indeed improve the average chase time and the total broadcast bandwidth required.*

Keywords: *MANETs, Routing protocols, Route Discovery, Control Packets, Chase Packets, Performance analysis.*

1. Introduction

Traditional wired networks have been the main focus of research. However, they are useful but not suitable for mobile situations. When mobile devices such as notebooks and PDAs appeared, users wanted wireless connectivity, and this duly become a reality. Wireless networks may be infrastructure-oriented, as in conventional WiFi [11] or infrastructure-less as in Mobile Ad-hoc Networks (MANETs) [3, 13, 17]. One of the dominant initial motivations for MANET technology came from military applications in environments where there is no infrastructure. However, while such applications remain important, MANET research has diversified into areas such as disaster relief, sensors networks, and personal area networks [17].

The design of an efficient routing strategy is particularly challenging in MANETs due to its limited resources [11]. Many multi-hop routing protocols have been proposed and investigated in the literature [1, 2, 6, 13, 14] Existing routing protocols can be classified into three categories: proactive (table-driven), reactive (on-demand), and hybrid [1]. Proactive routing protocols determine routes to all the destinations (or parts of the network) statically at start up and these are then maintained via a periodic route update process. An example of this class of routing protocols is the Optimized Link State Routing Protocol (OLSR) [2]. However, in on-demand routing, routes are determined dynamically when required by the source using a route discovery process as in Dynamic Source Routing (DSR) [7]. This approach uses lower bandwidth and lower power consumption, since nodes have no associated periodic tasks, a very appealing characteristic in a MANET scenario [1]. When a source needs to send messages to a destination, it initiates a broadcast-based route discovery process to look for one or more possible paths to the destination. Hybrid routing protocols are both reactive and proactive in nature The Zone Routing Protocol (ZRP) [6] is an example belonging to this class.

Despite the fact that significant effort has been devoted to minimising the overhead of the route discovery process, most existing strategies that have been proposed in the literature [4, 5, 15, 18] have one or more of the following deficiencies.

- They may need special resources such as location determining technology (e.g. GPS) which might be costly or hard to implement. An example of such strategies is presented in [18].
- They may require storage of a large amount of historical data in the nodes which requires additional memory and drains power. For some examples see [4, 10, 15].
- They may depend on information that has a high chance of being stale as is the case with some examples in [10, 15].
- They do not improve the cashing and overhearing capabilities, i.e. information gathering, of new routes. Some of such strategies are presented in [9, 10, 12, 15, 18].
- They may use small portion of the channel bandwidth for broadcasting route requests which will delay the discovery process and/or reduce chances of finding new routes. Such delays in route discovery may also increase the possibility of losing these routes after their discovery. For more details and examples see [5].
- They may not utilize the full bandwidth for sending chase packets which are small control packets to stop the propagation of the fulfilled route requests [5, 19]. Such lack of utilization may hinder the chasing process. For an example see [5].
- They may flood the network with broadcast messages causing a broadcast storm that waste both bandwidth and power as discussed in [8].

Chase packets have been proposed to minimise the route discovery overhead [5, 19]. In this paper, we will propose a new algorithm that uses the chase packet approach in a more efficient way by using a dual tier approach. The rest of the paper of the paper is organised as follows. Section 2 will introduce some of the notation and definitions. Section 3 presents the related work in the literature while Section 4 presents our newly proposed algorithm then evaluates its performance and conducts a comparative study demonstrating the superiority of our algorithm over others in the literature for example [5]. Finally, Section 5 concludes this study.

2. Related Work

An algorithm for route discovery that eliminates the need for historical or location information has been proposed in [5]. It achieves this by employing a chase packet to dampen the route request propagation after finding the needed route. The algorithm works by broadcasting a *route request* using a fraction of the channel bandwidth while the rest of that bandwidth is dedicated to transmit the *route reply* and broadcast the chase packets after the route is found. Figure 1, illustrates the algorithm using: channel 1 which, by supposition, takes ¼ of the channel time to transmit the *route request*; and channel 2 which is allocated the rest of the channel time (¾) to transmit the chase packet or the *route reply*. The ratio 1:4 was used in [5]. The main deficiency in this approach is that it favours the chase packet over the *route requests* delaying *route requests* sent by the source node and other nodes in the network. Moreover, throughout the network, all *route requests* will suffer a slowdown due to the fact that, regardless of the source, all chase packets are favoured, over the *route request* messages, by the routing algorithm.

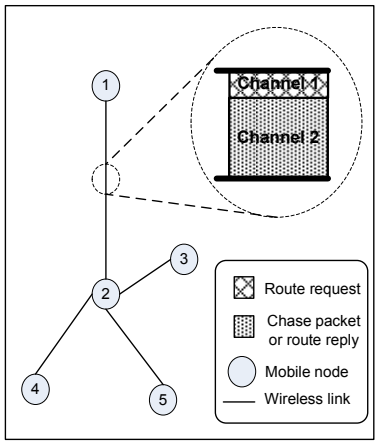

Figure 1. Division between chase packet and route reply or route request

In this algorithm, the sender is the responsible for initiating the chase packet which may then experience excessive delay in catching up with the *route request*. This shortcoming has been addressed in [19] which proposed that chase packets be sent by every node involved in discovering the route rather than merely the source node itself. However, this algorithm may congest the network with traffic and cause a storm of chase packets. Moreover, it is unsuitable for multi-path discovery.

In the rest of the paper, we will propose a new route discovery algorithm that overcomes some of the afore-mentioned deficiencies.

3. Preliminaries

Let $N = \{n_1, n_2, \cdots, n_l\}$ be the set of nodes in some network of diameter, *d*, where the diameter of MANET is the path with the smallest number of hops between the furthest two arbitrary nodes in the network [16]. Let $s \in N$ be a source node and define a function, $h_s : N \to Z^+ \cup \{0\}$ where $h_s(v)$ is the hop count between s and some other node $v \in N$. Let *m* and *n* be two positive integers with $0 \le m \le n \le d$. The subset of all nodes, $v \in N$, for which $m < h_s(v) \le n$ will be called a *tier* of the network with respect to *s*. A sequence of positive integers α_i where $0 \le i \le k$, $\exists\ \alpha_0 = 0$, $\alpha_k = d$ and $\alpha_{i-1} < \alpha_i\ \forall i, 0 < i \le k$ defines a set of disjoint tiers of *N* with respect to *s*. In general, the tier τ_i is the subset of all nodes, $v \in N$, for which $\alpha_{i-1} < h_s(v) \le \alpha_i$. The width of τ_i is equal to $\alpha_i - \alpha_{i-1}$, $\forall i, 0 < i \le k$.

4. The proposed algorithm

We propose a new route discovery algorithm that also uses the chase packet approach and a multi-tier approach. The algorithm works by establishing a *neighbourhood* that includes most of the likely destinations for the source node on hand and which, with its complement, the "*beyond neighbourhood*" forms a dual-tier partition of the network. The idea is to process the route requests using the full channel bandwidth during the first phase, within the *neighbourhood*. However, a slight delay is introduced to slowdown the propagation of route requests within the *beyond neighbourhood* region. As soon as the source receives a *route reply* message, it will broadcast a chase packet using the full channel bandwidth, in an attempt to terminate propagation of the route request. This chase packet travels faster than the route request message in the *beyond neighbourhood* region.

Formally, using multi tiers, let us consider the tier-partition $\{\tau_1, \tau_2\}$ where τ_1 and τ_2 are the *neighbourhood* and *beyond neighbourhood* tiers respectively. It is obvious that the tiers are disjoint sets so $\tau_1 \bigcap \tau_2 = \phi$. Let us consider a source node s, so that any node $v \in \tau_1$ satisfies the condition $\alpha_0 < h_s(v) \leq \alpha_1$ and any node $u \in \tau_2$ should satisfy the condition $\alpha_1 < h_s(u) \leq \alpha_2$ the values for the tiers boundaries are as follows: α_0 = 0, α_1 should satisfy the condition $\alpha_0 < \alpha_1 < \alpha_2$ and α_2 = the network *diameter*. The α_1 can be set to any initial value depending on the environment and will be continuously tuned to adapt to the current situation using the values of $h_s(f)$ for all f with respect to s.

Figures 2, 3, and 4 show the steps that will be performed by each node upon receiving a *route request, route reply*, or chase packet respectively.

```
Steps preformed by each node upon receiving
a route request in 2-tier approach
If chase packet has been received then
    Discard the route request
Else
    If hop_count <= α1 then
        Broadcast the route request.
    Else
        Wait t time
        Broadcast the route request
    End if
End if
```

Figure 2. Route request messages processing at each node

```
Steps performed by each node upon receiving
the route reply message
If current node is the sender then
            Create a chase packet
            Broadcast the chase packet
            Start transmitting the data
Else
            Route the route reply message to the sender.
End if
```

Figure 3. Route reply message processing at each node.

```
Steps performed by each node upon receiving
the chase packet

If the chase packet is a duplicate then
        Discard it.
 Else
        If the route request has been received then
            If the route request has been broadcasted then
                Broadcast the chase packet.
            Else
                Discard the route request and chase packet.
            End if
        Else
             Discard the chase packet.
       End if
 End if
```

Figure 4. Chase packets processing at each node.

5. Performance evaluation

We assume that all nodes are identical, most of the destinations are within the 1st tier, τ_1, and the links are bidirectional. In this algorithm, soon as a node f, at distance $h_s(f)$ from the source s, finds a route; it will send a *route reply* to the source, in the reverse direction and the f will then stop propagating the *route request*. However, other nodes may continue to propagate the route request throughout the network since they may not be aware of the successful route discovery by node f.

When the source receives the route reply, it will initiate and broadcast the chase packet while the route request would still be propagating throughout the network at almost twice the distance from the source i.e. $2h_s(f)$. Moreover, by the time the chase packet is $2h_s(f)$ distance from the source, the route request would have propagated further and the chase packet would still be chasing it.

Let us consider the *velocity* of the chase packet v_2 and the *velocity* of the route request to be v_{11} within τ_1 and v_{12} within τ_2. Within the 1st tier, τ_1, the route request and the route reply as well as the chase packet will all be travelling using the full channel time, i.e. $v_2 = 1$. Thus:

$$v_{11} = v_2 \tag{1}$$

On the other hand, within the 2nd tier, τ_2, the algorithm will slowdown the propagation of the route request from v_2 to $v_2\beta$, where $\beta = \alpha_1 / \alpha_2$, to allow the chase packet to catch the route request. So the new velocity, v_{12}, will be:

$$v_{12} = v_2\beta \tag{2}$$

When the chase packet is initiated we have a distance of $2\,h_s(f)$ between the route request and the chase packet. Moreover, the chase packet will always catch the route request in τ_2 where $v_{12} < v_2$.

Below we will calculate the chase time, t_c, and the total time, *T*, considering the following two possibilities:

- Case 1: The route request will be in the 1st tier, τ_1, when the source initiates and broadcast the chase packet. i.e. $h_s(f) \le \alpha_1 / 2$
- Case 2: The route request will be in the 2nd tier, τ_2, when the source initiates and broadcast the chase packet. i.e. $h_s(f) > \alpha_1 / 2$

5.1 Calculating the Chase time t_c

The time t_c that is needed for the chase packet to catch the request can be calculated from the following formula:

$$v_2 t_c = \begin{cases} \alpha_1 + (t_c - (\alpha_1 - 2h_s(f))/v_2)v_{12} & h_s(f) \le \alpha_1 / 2 \\ 2h_s(f) + v_{12} t_c & h_s(f) > \alpha_1 / 2 \end{cases} \quad (3)$$

Let us consider the Case 1. We can use (2) and (3) to calculate the value of chase time t_c from the following formula:

$$v_2 t_c = \alpha_1 - \alpha_1 \beta + 2h_s(f)\beta + v_2 t_c \beta \quad (4)$$

Giving us the value for the chase time t_c as follows:

$$t_c = \alpha_1 + \beta(2h_s(f) - \alpha_1)/v_2(1-\beta) \quad (5)$$

Now let us consider Case 2. From (2) and (3) we can conclude the following formula:

$$v_2 t_c = 2h_s(f) + v_2 t_c \beta \quad (6)$$

Then the chase time t_c can be calculated as follows:

$$t_c = 2h_s(f)/v_2(1-\beta) \quad (7)$$

5.2 Calculating the total broadcast time *T*

The total broadcast time, *T*, is the time from sending a particular route request until the chase packet catches such route request. It can be calculated as:

$$T = 2h_s(f)/v \ + t_c \quad (8)$$

Where v stands for the route request velocity. In the dual tier approach the value of v differs to be v_{11} or v_{12} depending on whether the route request is currently broadcasted within the tier τ_1 or τ_2 respectively.

For the Case 1, the total broadcast time, *T*, will be calculated as:

$$T = 2h_s(f)/v_{11} + t_c \quad (9)$$

By taking the value of t_c from (5) and simplifying the formula we get:

$$T = \alpha_1 + (2h_s(f) - \beta\alpha_1)/v_2(1-\beta) \quad (10)$$

For the Case 2, the total broadcast time, *T*, can be calculated as follows:

$$T = \alpha_1 / v_{11} + (2h_s(f) - \alpha_1)/v_{12} + t_c \quad (11)$$

Using (1) and (2) in (11), *T* can be simplified as:

$$T = (\alpha_1\beta + 2h_s(f) - \alpha_1)/v_2\beta + t_c \quad (12)$$

By taking the value of t_c from (7) we get:

$$T = (\alpha_1\beta + 2h_s(f) - \alpha_1)/v_2\beta + 2h_s(f)/v_2(1-\beta) \quad (13)$$

5.3 Comparison with the existing algorithm

In this section we conduct a comparison between our new algorithm and the algorithm proposed in [5]. Such comparison aims at studying the behaviour of both algorithms and evaluating the growth of the chase time, t_c, and the total broadcast time, *T*, using various values for the hop count from the source to the finder, $h_s(f)$ for different finders under the same environment. Figures 5 and 6 show the chase time, t_c , and the total broadcast time, *T*, respectively using various hop count values. These figures are drawn using the Tables 1 and 2 respectively and using the values $\alpha_1 = 15$ and $\alpha_2 = 60$ which will yield a delay of 0.75 which corresponds to the same delay in [5].

In figure 5, we can see that our new algorithm will always need much less chase time to catch the route request message. Moreover, as the value of the hop count increases the time our algorithm requires to catch the route request will be much less as compared to the time required by the old algorithm in [5]. For example, when the hop count, $h_s(f)$, was given the values 10 and 20 the chase times needed were 26.67 and 53.33 for our algorithm but 40 and 80 for the old algorithm with chase time differences of 13.33 and 26.67 respectively.

Hop count $h_s(f)$	Chase Time t_c	
	New	Old
8	21.33	32
10	26.67	40
12	32.00	48
14	37.33	56
16	42.67	64
18	48.00	72
20	53.33	80
22	58.67	88
24	64.00	96
26	69.33	104
28	74.67	112

Table 1. The chase time for various hop counts

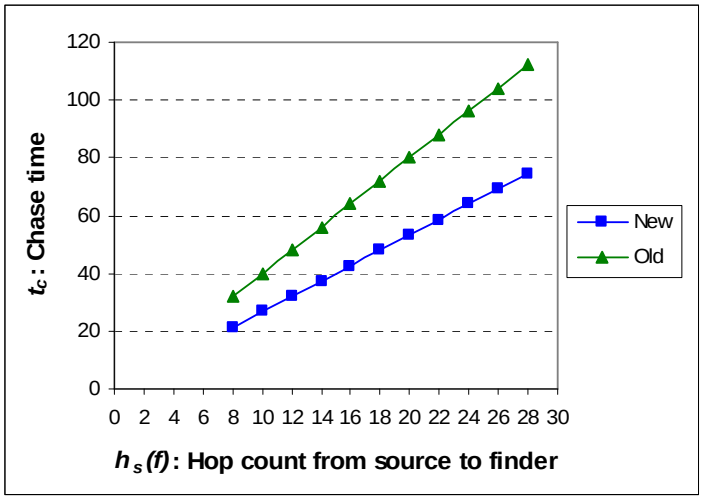

Figure 5. The performance for the old and new route discovery algorithms using the chase time.

For the total broadcast time, *T*, figure 6 shows the performance of our algorithm compared to the algorithm in [5] using different hop count values. We can clearly see that for the hop count values 10 and 20 our algorithm needed total times of 61.67 and 168.33 but the old algorithm needed total times of 120 and 240 respectively with total broadcast time differences of 58.33 and 71.67 respectively.

Hop count $h_s(f)$	Total Time T	
	New	Old
8	40.33	96
10	61.67	120
12	83.00	144
14	104.33	168
16	125.67	192
18	147.00	216
20	168.33	240
22	189.67	264
24	211.00	288
26	232.33	312
28	253.67	336

Table 2. The total broadcast time for various hop counts.

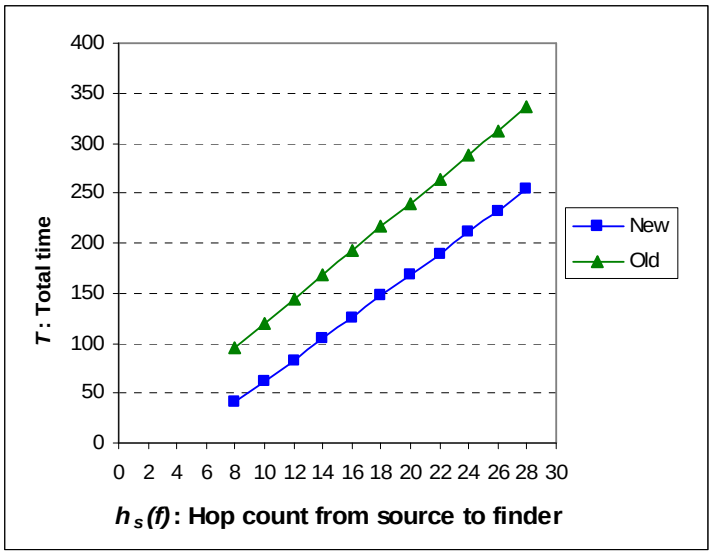

Figure 6. The performance for the old and new route discovery algorithms using the total time.

Comparing the chase time and the total broadcast time for our new route discovery algorithm with the same for the existing algorithm in [5] as shown in figure 5 and figure 6. We can clearly see the superiority of our algorithm for both times, i.e. less latency. Furthermore, looking at the asymptotic behaviour, our algorithm will further outperform the existing algorithm with further growth the hop count.

6. Conclusion

In this paper, we have developed a new dual tier route discovery algorithm for MANETs with chase packets that works by including most of the likely destinations for the source node on hand in the 1st tier and broadcasting the route requests with full channel time within such tier. Also, to provide a much better chance for the chase packets to catch the route requests afterwards, the algorithm delays the propagation of the route requests within the 2nd tier which will in turn minimise the network congestion. The algorithm is also adaptive and continuously updates the boundary between the two tiers to provide the best performance. Moreover, we have provided a detailed performance evaluation for our new algorithm and compared it with the existing algorithm in the literature [5]. Our evaluation showed that our algorithm will require less chase time and less total broadcast time which makes it superiority.

References:

1. Abolhasan, M., T. Wysocki, and E. Dutkiewicz, *A review of routing protocols for mobile ad hoc networks.* Ad Hoc Networks. **2**(1): p. 1-22, 2004.
2. Adjih, C., et al., *Optimized Link State Routing Protocol*, The Internet Engineering Task Force, IETF, RFC 3626, , 2003.
3. Chlamtac, I., M. Conti, and J.J.N. Liu, *Mobile ad hoc networking: imperatives and challenges.* Ad Hoc Networks. **1**(1): p. 13-64, 2003.
4. Das, S.R., et al. *Comparative performance evaluation of routing protocols for mobile, ad hoc networks*, 1998.

5. Gargano, L., M. Hammar, and A. Pagh. *Limiting flooding expenses in on-demand source-initiated protocols for mobile wireless networks*. in *18th International Parallel and Distributed Processing Symposium*, 2004.
6. Haas, Z.J., M.R. Pearlman, and P. Samar, *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*, IETF MANET Working Group, 2002.
7. Johnson, D., D. Maltz, and Y.-C. Hu, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, The Internet Engineering Task Force, IETF, draft-ietf-manet-dsr-09.txt, 2003.
8. Katia, O., V. Kumar, and T. Gene, *Flooding for reliable multicast in multi-hop ad hoc networks.* Wireless Networks. **7**(6): p. 627-34, 2001.
9. Koutsonikolas, D., et al. *On optimal TTL sequence-based route discovery in MANETs*. in *Second International Workshop on Wireless Ad Hoc Networking*, 2005.
10. Minematsu, M., et al. *HOWL: an efficient route discovery scheme using routing history in ad hoc networks*. in *27th Annual IEEE International Conference on Local Computer Networks (LCN'02)* 2002.
11. Murthy, S. and M. B., *Ad Hoc Wireless Networks: Architectures and Protocols*: Prentice Hall, 2004.
12. Nicholas, C. and L. Mingyan, *Revisiting the TTL-based controlled flooding search: optimality and randomization*, in *Proceedings of the 10th annual international conference on Mobile computing and networking*, ACM Press: Philadelphia, PA, USA, 2004.
13. Perkins, C., *Introduction to Ad hoc networking* Addison Wesley, 2001.
14. Ramanathan, S. and S. Martha, *A survey of routing techniques for mobile communications networks.* Mob. Netw. Appl. **1**(2): p. 89-104, 1996.
15. Robert, C. and R. Samir, *Query localization techniques for on-demand routing protocols in ad hoc networks*, in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ACM Press: Seattle, Washington, United States, 1999.
16. Stuart, K., C. Tracy, and C. Michael, *MANET simulation studies: the incredibles.* SIGMOBILE Mob. Comput. Commun. Rev. **9**(4): p. 50-61, 2005.
17. Tanenbaum, A., *Computer Networks*: Pearson Education, 2003.
18. Young-Bae, K. and H.V. Nitin, *Location-aided routing (LAR) in mobile ad hoc networks.* Wireless Networks. **6**(4): p. 307-21, 2000.
19. Zhang, H. and Z.P. Jiang. *On reducing broadcast expenses in ad hoc route discovery*. in *Second International Workshop on Wireless Ad Hoc Networking (WWAN) ICDCSW'05)*, 2005.