UCL DEPARTMENT OF
COMPUTER SCIENCE

Research Note
RN/07/05

Adaptive Proactive Routing Algorithm for Mobile Ad Hoc Networks

01/03/2007

Yangcheng Huang

Saleem Bhatti

Søren-Aksel Sørensen

Abstract

Proactive MANET routing protocols tend to provide smaller route discovery latency than on-demand protocols because they maintain route information to all the nodes in the network at all time. However, the downside for such protocols is the excessive routing control overhead that is generated by disseminating periodic HELLO messages and topology control messages. Due to the resource-constrained nature of wireless networks, the routing overhead increases channel contention, leads to network congestions and lowers significantly network performance. In order to mitigate the side effects of the soft update control overheads, we propose two adaptive proactive routing algorithms, namely DT MIAD and DT ODPU. By tuning the value of refresh intervals dynamically and automatically, refresh updates are triggered based on traffic conditions and node mobility. We have shown through simulations that, the proposed adaptive routing algorithm outperforms traditional proactive routing protocols like OLSR.

Adaptive Proactive Routing Algorithm for Mobile Ad Hoc Networks

Yangcheng Huang
Department of Computer Science
University College London
Gower Street, London WC1E 6BT
Email: y.huang@cs.ucl.ac.uk

Saleem Bhatti
School of Computer Science
University of St. Andrews
St. Andrews, UK
Email: saleem@dcs.st-and.ac.uk

Søren-Aksel Sørensen
Department of Computer Science
University College London
Gower Street, London WC1E 6BT
Email: S.Sorensen@cs.ucl.ac.uk

***Abstract*— Proactive MANET routing protocols tend to provide smaller route discovery latency than on-demand protocols because they maintain route information to all the nodes in the network at all time. However, the downside for such protocols is the excessive routing control overhead which is generated by disseminating periodic HELLO messages and topology control messages. Due to the resource-constrained nature of wireless networks, the routing overhead increases channel contention, leads to network congestions and lowers significantly network performance. In order to mitigate the side effects of the soft update control overheads, we propose two adaptive proactive routing algorithms, namely *DT_MIAD* and *DT_ODPU*. By tuning the value of refresh intervals dynamically and automatically, refresh updates are triggered based on traffic conditions and node mobility. We have shown through simulations that, the proposed adaptive routing algorithm outperforms traditional proactive routing protocols like OLSR.**

I. Introduction

Mobile ad hoc networks (MANETs) are characterized with frequent topology changes and resource constrains (such as battery life and bandwidth). Typical MANET applications, including emergency rescue operations, and battlefield communications, exhibit high degrees of connection dynamics due to mobility and complex natural environments (thunderstorms etc). Nodes may join the network at any time, and leave the network when they run out of power, or move out of the reachability of radio. Link characteristics, such as bandwidth, change frequently due to signal interference and radio propagation fading. Consequently, a fundamental challenge in ad hoc networks is the design of routing protocols that can respond quickly to network changes.

Proactive protocols like OLSR [1], TBRPF [2] and DSDV[3] tend to provide smaller route discovery latency than on-demand protocols like AODV[4] and DSR[5] because they maintain route information to all the nodes in the network at all time. However, the downside for such protocols is the excessive routing control overhead generated by disseminating periodic HELLO messages and topology control messages in state maintenance. Due to the resource-constrained nature, proactive routing algorithms have a fundamental trade-off between the performance and the routing overhead. That is, although a small refresh interval could speed up adaptation to network conditions, the overhead introduced might cause channel congestions and lower network performance.

In order to mitigate the side effects of the soft update control overheads, we propose adaptive proactive routing algorithms to adjust refresh intervals of proactive routing protocols according to node mobility. Essentially such adaptive algorithms are feedback based. The protocols' behaviors (i.e. parameters) are tuned according to the status of hosting environments, such as available bandwidth and channel loss rate, in order to achieve better performance with less control overhead. Therefore, there are two major issues: how to sense network changes and how to tune the parameters of the protocols.

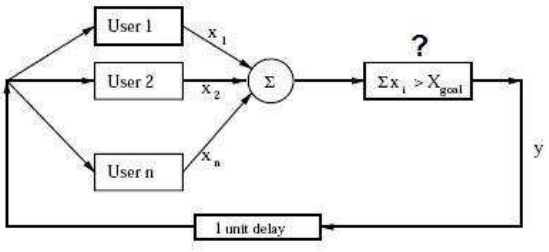


Fig. 1. Traditional Feedback Based Control Model

Up till now, there have been several adaptive routing approaches for MANETs [6] [7] [8]. Benzaid et al[6] presented an approach to adjust refresh frequency based on node mobility and the MPR status of its neighboring nodes. Ramasubramanian et al[7] proposed a zone-based hybrid routing algorithm which combined proactive and reactive strategies. Boppana et al[8] proposed an adaptive Distance Vector routing algorithm by adopting flexible route update strategies according to conditions. We contend that, although well designed, these adaptive approaches have the following problems.

First, dependency on network measurement. The routing performance of the approaches[6] [7] largely depends on the accuracy of network measurement. Due to network dynamics, it is still an open question on how to get accurate estimation of real-time network/traffic characteristics in practice. In consequence, the applicability of these algorithms might be jeopardized.

Second, increased complexity. For example, in [7], the operations in zone maintenance and continuous network monitoring not only introduce extra processing overhead but also

increase the complexity in configuration and implementation. The performance of ADV algorithm[8] is determined by *constant trigger thresholds*, which need to be manually configured.

Third, unbounded performance. For example, in ADV algorithm[8], the route update frequency increases quickly with node mobility, which brings larger overheads than periodic updates. Also, since only partial route information is maintained, ADV takes longer for a new connection to find a valid route.

In order to solve these problems, this paper proposes two adaptive proactive routing algorithms, namely *DT_MIAD* and *DT_ODPU*. By tuning the value of refresh intervals of soft-state timers *dynamically* and *automatically*, the refresh updates are triggered based on network load and mobility conditions. We have shown with simulations that, the proposed adaptive routing algorithm outperforms traditional proactive routing protocols like OLSR.

Compared with existing algorithms, our approach shows the following benefits.

First, the operations of the proposed algorithms are independent of network measurement and node mobility detection. Based on analytical studies on link change rate, we propose a simple method in detecting node mobility.

Second, the proposed algorithms are simple in both configuration and implementation. The adaptability process is totally automated with only a few parameters. Enlightened by the feedback based control theory, the proposed algorithm can be implemented incrementally, with no need to make significant changes to the existing protocols.

Overall, this paper makes the following contributions. First, it presents novel adaptive routing algorithms that provide performance guarantee. Second, it proposes a new method in sensing network dynamics, which is independent of network measurement techniques. Third, it introduces a Multiplicative-Increase Additive-Decrease (MIAD) controller to adjust soft-state refresh rate to the conditions of node mobility and data traffic.

The rest of the paper is organized as follows. Section 2 gives some background information on traditional proactive routing algorithms. A formal problem definition is presented in section 3. Section 4 gives the detailed description of the routing algorithms. Section 5 introduces the simulation configurations used in this study. Section 6 presents our observations based on the NS2 simulations. Related work is listed in section 7 and conclusions are summarized in section 8.

II. Proactive Routing Protocols for MANETs

In this section, we present an overview of the traditional proactive routing algorithms including Link State algorithm such as OLSR and Distance Vector algorithm such as DSDV. Then we explain the motivations of tuning HELLO intervals in the proposed algorithms.

A. Traditional Proactive Routing: LS vs. DV

In Link State(LS) protocols like OLSR[1], each node discovers and maintains a complete and consistent view of the network topology, by which each node computes a shortest path tree with itself as the root (i.e. *SPF* algorithm), and applies the results to build its forwarding table. This assures that packets are forwarded along the shortest paths to their destinations.

LS protocols rely on periodic refresh messages to reflect topology changes and maintain correct topology information. Each node sends HELLO messages periodically to discover new neighbors and detect link failures. Unlike LS protocols such as OSPF, in which the topology update is triggered by network change events, LS protocols in MANETs advocate periodic topology update strategy, in order to avoid the large amount of topology update messages triggered by frequent topology change events.

In Distance Vector(DV) protocols like DSDV[3], each node maintains a routing table containing the distance from itself to all other nodes in the network. Each node broadcasts periodically its routing table to each of its neighbors and uses similar routing tables from neighboring nodes to update its table. The route selection is based on Distributed Bellman-Ford(DBF) algorithm. To keep up with network changes, DV algorithms use both periodic and triggered updates.

The main problem of traditional proactive routing (especially LS algorithm) lies in the use of fixed timer intervals. The refresh intervals are configured by administrators, usually with the default values recommended by protocol designers. Basically, high mobility demands small intervals to speedup failure detection, while low mobility only needs relatively large intervals to reduce control overhead. Due to the non-uniform distribution of node mobility, both temporally and spirally, the fixed timer intervals fail to be effective when/where the node mobility is high and efficient when/where the node mobility is low. Thus, the refresh intervals need to be adapted to network conditions.

B. Impacts of Neighbor Sensing and Topology Update

The adaptive algorithm proposed is based on our recent studies on soft-time performance in Mobile Ad hoc networks.

It is commonly believed that a smaller update/refresh interval in LS routing protocols could speed up adaptation to changes at the expense of increased overhead. However, there has been little research on how the refresh intervals impact routing performance.

In [9], we investigate the effects of tuning refresh intervals on proactive routing performance. We compare the difference in performance when changing the intervals for HELLO and topology update messages of OLSR. We find that, although reducing refresh intervals could improve routing performance, the intervals for some message types (HELLO messages) have a bigger impact on proactive routing performance than for other message types (topology update messages). We give a further discussion of the impacts in [10].

Based on these observations, we put the focus of the adaptability study on neighbor sensing mechanisms, rather than on topology update. That is, since HELLO intervals have a significant impact, it can be reasonably inferred that, the

adaptability of the routing protocol can be achieved by adapting/tuning neighbor update process to network conditions.

III. Problem Formalization

In this section, we present a formal definition of adaptive routing problems, followed by analysis on the corresponding performance boundaries in terms of link detection latency and control overhead.

A. Definition

Let r be the refresh rate. Let $G(r)$ be the quantitative relationship between refresh rate $r \in \phi$ and routing throughput G(i.e. Gain). Let $C(r)$ be the quantitative relationship between refresh intervals $r \in \phi$ and routing overhead C(i.e. Cost). Based on the studies on the impacts of HELLO refresh rate on routing performance, we can infer that, given that other factors (such as velocity $v \in \gamma$) be the same,

$$G(r_1) \leq G(r_2), where r_1 \leq r_2$$
$$C(r_1) \leq C(r_2), where r_1 \leq r_2$$

Therefore, we expect to find a function f, satisfying

$$\exists f, \forall r_1, r_2 \in \phi, r_1 \leq r_2,$$
$$r(t) = f(v, \mu, \rho, t),$$

(1)

$$\forall v \in \gamma,$$
$$r_v(t) \leq r_2$$
$$\overline{G_v(r)} \leq \overline{G_v(r_2)}$$
$$r_1 \leq r_v(t)$$
$$\overline{C_v(r_1)} \leq \overline{C_v(r)}$$

(2)

$$\forall v_1, v_2 \in \gamma, v_1 \leq v_2,$$
$$\left|\overline{G_{v_1}(r)} - \overline{G_{v_2}(r)}\right| \leq \left|\overline{G_{v_1}(r_1)} - \overline{G_{v_2}(r_1)}\right|$$
$$\left|\overline{G_{v_1}(r)} - \overline{G_{v_2}(r)}\right| \leq \left|\overline{G_{v_1}(r_2)} - \overline{G_{v_2}(r_2)}\right|$$

B. Bounded Link Detection Latency

Link change events include link breakage and link establishment. First, let's look at the link failure detection latency.

We assume that the arrival of a link failure event is an independent, identically distributed Poisson process with arrival rate λ. The assumption is reasonable, if the node degree is small and the nodes are moving randomly so that the process of link breakage is totally random.

Consider an arbitrary period, starting at t_0. Let L_b be the link breakage detection latency of the proactive neighbor sensing mechanism. Let X_b be the time of $first$ link change occurrence after t_0. Let $m * \frac{1}{r(t_0)}$ be the link state time-out interval at t_0 (i.e. m times of HELLO interval), which is used by soft-state timer to detect link breakage.

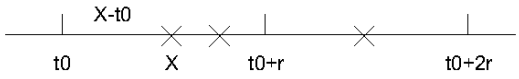


Fig. 2. Soft State based Failure Detection

Then the link breakage detection latency of A at t_0 can be approximated by,

$$L_b = t_0 + m * \frac{1}{r(t_0)} - X = \frac{m}{r(t_0)} - (X_b - t_0) \quad (1)$$

According to the assumption,

$$X_b - t_0 \propto exp(\lambda) \quad (2)$$

Therefore, the expected link detection latency of A at t_0 therefore can be approximated by,

$$l_b = E(L_b) = \frac{m}{r(t_0)} - E(X_b - t_0) = \frac{m}{r(t_0)} - \frac{1}{\lambda} \quad (3)$$

Given $r_1 \leq r(t_0)$,

$$l_b = \frac{m}{r(t_0)} - \frac{1}{\lambda} \leq \frac{m}{r_1} - \frac{1}{\lambda} \quad (4)$$

Therefore, the link failure detection latency is bounded by l_{b,r_1}.

Similarly, the link discovery latency (i.e. the delay in discovering new established links) is bounded by $l_{e,r1}$.

C. Bounded Control Overhead

One lesson from existing adaptive routing algorithm is that the control overhead *must* be bounded. That is, the control overhead should not increase linearly with network conditions.

Consider an arbitrary period, starting at t_0. The control overhead generated during the period [t_0, $t_0 + \delta t$] could be approximated by $C(t_0) = r(t_0) * \delta t$. The overall control overhead could be approximated by

$$C = \sum^{T} r(t)\delta t \doteq \int_0^T r(t)dt$$

Given $r \leq r_2$,

$$C \leq \int_0^T r_2 dt = r_2 T$$

IV. Adaptive Proactive Routing Protocol

In this study, we improve periodic update strategies of existing proactive routing protocols by adapting dynamically refresh rates to neighbor changes. The proposed method inherits simplicity and robustness from traditional soft-state mechanism. On the other hand, the adaptability to mobility helps achieve the balance between performance and overhead. In the following paragraphs, we present the details of our proposed algorithms, namely *DT_MIAD* (Dynamic Timer Based on Multi-Increase Additive Decrease) and *DT_ODPU* (Dynamic Timer Based on On-Demand Proactive Update).

A. Dynamic Timer Based on Multi-Increase Additive Decrease

The dynamic timer algorithm based on MIAD is inspired by control-theoretic adaptive mechanisms similar to those widely adopted in the Internet, i.e. Additive Increase Multiplicative Decrease(AIMD) of TCP, which is used to adjust sending rates in response to network congestions: the sending rate of TCP in congestion avoidance state is controlled by a congestion window which is halved for every window of data containing a packet drop, and increased by one packet per window of data acknowledged. Our approach in this algorithm uses a Multiplicative-Increase Additive-Decrease (MIAD) controller to adapt the soft-state refresh rate r to the conditions of node mobility and data traffic.

Briefly, refresh rate r is multiplied by factor α ($\alpha > 1$) if node mobility or data packet drop rate increases, and otherwise decremented by factor β. By aggressively increasing r in presence of rise of packet failure rate and network change rate, the routing algorithm improves link detection process, which reduces packet drops and increases link availability. Whenever link change rate descends, the routing algorithm lowers refresh frequency conservatively which finally reaches a steady state.

Therefore, the key question is, what is the quantitative relationship between node mobility and the link change rate? If it is linear, the node mobility can be simply detected by monitoring the link change rate. We clarify this issue in the following paragraphs and present the details of the proposed algorithm.

Any change in the set of links of a node may be either due to the arrival of a new link or due to the breaking of a currently active link. Thus, the expected link change rate for a node ψ is equal to the sum of the expected new link arrival rate η and the expected link breakage rate ξ.

Prince Samar and Stephen B. Wicker studied the theoretical quantitative relationship between link change rate ψ and factors including node velocity in [11]. They found that, in a practical ad hoc or sensor network where "the number of neighbors of a node is bounded", the expected rate of link breakages ξ is equal to the expected rate of new link arrivals η. Therefore, the expected link change rate for a node ψ equals 2 times of the expected new link arrival rate η.

$$\dot{\psi(v)} = \dot{\eta(v)} + \dot{\xi(v)} = 2\dot{\eta(v)} \tag{5}$$

Equation (6) describes the expected new link arrival rate [11].

$$\dot{\eta(v)} = \frac{2R\delta}{\pi b}[\frac{v^2}{4}\int_0^{\pi} p(\phi)\log(\frac{b+\sqrt{b^2-v^2\sin^2\phi}}{v+v\cos\phi})d\phi + b^2\varepsilon(\frac{v}{b})] \tag{6}$$

Here, ε is the standard Complete Elliptic Integral of the Second Kind; ϕ is the direction of motion (i.e. the degree of the angle with x axis); $p(\phi)$ equals $1 + 3cos(2\phi)$; R is the transmission range; σ is the average density of nodes within a transmission zone; b is the maximum velocity.

TABLE I
DT_MIAD NOTATION

h_0	Initial HELLO interval of node i
$link\_chg\_cnt$	Change rate within current refresh period
$prev\_chg\_cnt$	Change rate within previous refresh period
$prev2\_chg\_cnt$	Change rate within the period before previous
β	The additive decrease rate
α	The multiplicative increase rate
h_{max}	The upper limit of refresh interval
h_{min}	The lower limit of refresh interval

Consider the impacts of node velocity v on link change rate ψ, i.e. the derivative of ψ with respect to v $\frac{d\psi}{dv}$. We obtain the following equations.

$$\dot{\psi_t'} > 0 \tag{7}$$

$$\dot{\psi_t''} > 0 \tag{8}$$

The relationship between link change rate and node velocity is shown intuitively in the Fig3. Here, the values of the parameters are set to a = 0m/s, b = 40m/s, R = 250 meters.

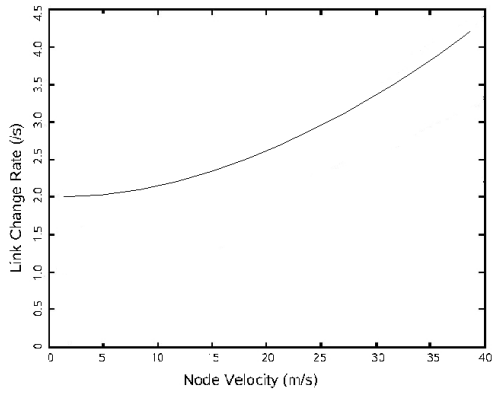


Fig. 3. Link Change Rate vs. Velocity

From Equation(7)(8), with the increase of node velocity, the expected link change rate increases. Moreover, the increasing speed of the expected link change rate increases with the node velocity. Therefore, we can examine the dynamics of link change rate in order to detect any changes of node mobility.

The pseudocode of the proposed algorithm is as shown in Algorithm 1. We use the notation as shown in Table I.

B. Dynamic Timer Based on On-Demand Proactive Update

Dynamic Timer Based on On-Demand Proactive Update (*DT_ODPU*) is based on the concept of *Finite State Machine*(FSM). The status of a node is roughly classified into two states: *dynamic* and *static*. When internal link changes are detected (link_chg_cnt $>$ 0), the node is in *dynamic* state; correspondingly, it uses a smaller refresh interval h_{min}. Otherwise, the node is still and uses a larger refresh interval h_{max}. In this algorithm, the state update is still *proactive* since refresh messages are still exchanged periodically; however, the

Algorithm 1 *DT_MIAD*

Input: $h_0 < \frac{1}{\beta}$
  $h \leftarrow h_0$
  $link\_chg\_cnt \leftarrow 0$
  $prev\_chg\_cnt \leftarrow 0$
  $prev2\_chg\_cnt \leftarrow 0$
  rest_of_init()
  loop
    Proporgate_Refresh_Msg()
    if $link\_chg\_cnt > prev\_chg\_cnt$ **then**
      if $link\_chg\_cnt - prev\_chg\_cnt > prev\_chg\_cnt - prev2\_chg\_cnt$ **then**
        $h \leftarrow decr\_h\_ival(h)$
      end if
    end if
    $h \leftarrow incr\_h\_ival(h)$
    $prev2\_chg\_cnt \leftarrow prev\_chg\_cnt$
    $prev\_chg\_cnt \leftarrow link\_chg\_cnt$
    $link\_chg\_cnt \leftarrow 0$
    DELAY(h)
    /*Performing other operations during the delay, including counting link changes, processing routing messages etc */
  end loop

  /* ———— decr_h_ival(h) ———— */
Input: $h > 0$
Output: $h \leftarrow decr\_h\_ival(h)$
  $h \leftarrow \frac{h}{\alpha}$
  if $h < h_{min}$ **then**
    $h \leftarrow h_{min}$
  end if
  SynchronizeTimerInterval()

  /* ———— incr_h_ival(h) ———— */
Input: $h > 0$ and $h_{max} < \frac{1}{\beta}$
Output: $h \leftarrow incr\_h\_ival(h)$
  $h \leftarrow \frac{h}{1-h*\beta}$
  if $h > h_{max}$ **then**
    $h \leftarrow h_{max}$
  end if
  SynchronizeTimerInterval()

refresh frequency(or refresh interval) is adjusted in *on-demand* manner.

The pseudocode of the proposed algorithm is as shown in Algorithm 2.

In the following paragraphs, we analyze the dynamics of refresh intervals under *DT_ODPU*.

We assume that the arrival of a link change event is an independent, identically distributed Poisson process with arrival rate λ. X_t is the number of occurrences in the interval $(0, t]$. Then we have

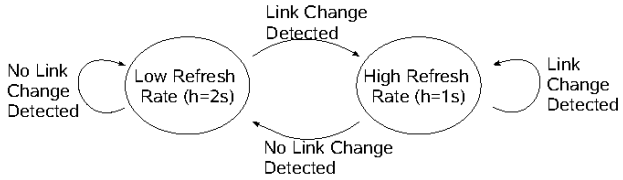


Fig. 4. State Transition Diagram of *DT_ODPU*

Algorithm 2 *DT_ODPU*

Input: $0 < h_{min} < h_{max}$
  $h \leftarrow h_{min}$
  $pre_refresh_time \leftarrow now$
  $link\_chg\_cnt \leftarrow 0$
  rest_of_init()
  loop
    if $link\_chg\_cnt > 0$ **then**
      Proporgate_Refresh_Msg()
    else if $now \geq (pre_refresh_time + h_{max})$ **then**
      Proporgate_Refresh_Msg()
      $pre_refresh_time \leftarrow now$
    end if
    $link\_chg\_cnt \leftarrow 0$
    DELAY(h)
    /*Performing other operations during the delay, including counting link changes, processing routing messages etc */
  end loop

$$P(X_t = 0) = e^{-\lambda t}$$

Let h_{min} be the lower limit of refresh interval and h_{max} be the upper limit. Thus the expected refresh interval can be approximated by,

$$h = \frac{1}{r} = \frac{1}{\sum_i r_i P_{r_i}} = \frac{1}{\frac{(1-e^{-\lambda h_{min}})}{h_{min}} + \frac{e^{-\lambda h_{max}}}{h_{max}}}$$

As shown in the Fig 5, the expected refresh interval reduces significantly when the link change rate gets increasing and converges gradually to its lower limit h_{min}. In addition, the larger the difference $h_{max} - h_{min}$ is, the better adaptability to network changes *DT_ODPU* would have.

V. Performance Analysis

A. Simulation Set-up

We implement the proposed algorithms in the OLSR implementation which runs in version 2.9 of NS2 [12] and uses the ad-hoc networking extensions provided by CMU [13]. The detailed configuration is shown in Table II and III.

The type of the wireless channel in this study is IEEE802.11 wireless LAN with distributed coordination function (DCF). The channel has a circular radio range with 250 meters radius

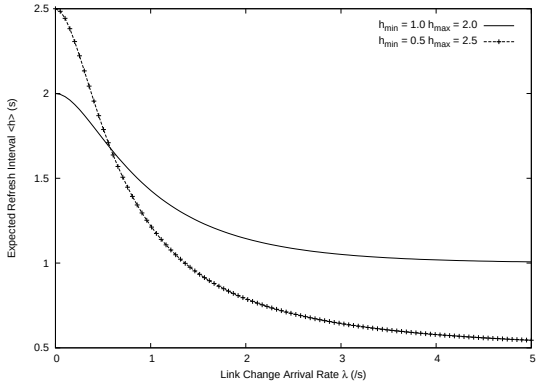


Fig. 5. Expected Refresh Interval vs. Link Change Rate

TABLE II
MAC/PHY LAYER CONFIGURATIONS

MAC Protocol	IEEE 802.11
Radio Propagation Type	TwoRayGround
Interface Queue Type	DropTailPriQueu
Antenna Model	OmniAntenna
Radio Radius	250m
Channel Capacity	2Mbits
Interface Queue Length	50

and a capacity of 2Mb/s. RTS/CTS (Request to Send / Clear To Send) mechanism is used to reduce frame collisions introduced by the hidden terminal problem and exposed node problem, which provides fairly reliable unicast communication between neighbors.

We use a network consisting of n nodes: $n = 20$ to simulate a low-density network, $n = 50$ to simulate a high-density network. Nodes are placed in a 1000 m^2 field. All simulations run for 100s.

We use the Random Trip Mobility Model, "a generic mobility model that generalizes random waypoint and random walk to realistic scenarios" [14] and performs perfect initialization. Unlike other random mobility models, Random Trip reaches a steady-state distribution without a long transient phase and there is no need to discard initial sets of observations. Manhattan Mobility Model is also used under different scenarios.

The mean node speed, v, ranges between 1m/s to 30m/s. For example, when the mean node speed is 20m/s the individual node speeds are uniformly distributed between 0m/s and 40m/s. The average node pause time is set to 5s.

A random distributed CBR (Constant Bit Rate) traffic model is used which allows every node in the network to be a potential traffic source and destination. The rate of each CBR traffic is 10kb/s. The CBR packet size is fixed at 512 bytes. There are at least $n/2$ data flows that cover almost every node.

For each sample point presented, 100 random mobility scenarios are generated. The simulation results are thereafter statistically presented with the mean of the metrics and the errors. This reduces the chances that the observations are dominated by a certain scenario which favors one protocol over another.

TABLE III
OLSR PARAMETERS

HELLO Interval	1s	2s
TC Interval	5s	5s
Neighbor Hold Interval	3s	6s
Topology Hold Interval	15s	15s

B. Performance Metrics

In each simulation, we measure each CBR flow's throughput and control traffic overhead and then calculate the mean performance of each metric as the result of the simulation.

Throughput is considered as the most straightforward metric for the MANET routing protocols[15]. It is computed as the amount of data transferred (in bytes) divided by the simulated data transfer time (the time interval from sending the first CBR packet to receiving the last CBR packet).

The control overhead consists of HELLO messages and TC messages. Considering the broadcasting nature of the control message delivery, the packets are counted by summing up the size of all the control packets $received$ by each node during the whole simulation period.

VI. OBSERVATIONS

In this section, we compare the routing performance of the proposed adaptive routing algorithms with that of standard proactive routing protocol, and present the observations under various factors, such as node velocity and node density.

A. Routing Performance under DT_MIAD

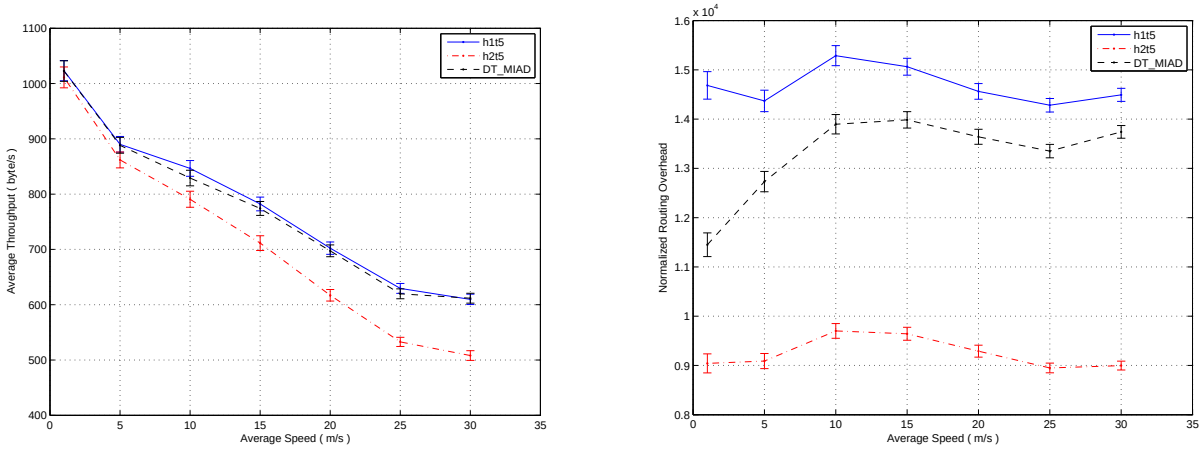
As shown in Fig 6 and Fig 7, OLSR with *DT_MIAD* achieves as good performance as standard OLSR with smaller interval($h = 1s$) but with much less overhead.

Further performance comparison with standard OLSR with larger interval($h = 2s$), OLSR with *DT_MIAD* shows good adaptability to node mobility. That is, with the increase of node mobility, the performance drop of OLSR with *DT_MIAD* is less significant. For example, as shown in Fig 7(a), when the node velocity increases from 10m/s to 20m/s, OLSR with *DT_MIAD* has 14.6% performance drop, while standard OLSR ($h = 2s$) has up to 32.6%. On the other hand, as shown in Fig 6(b), the overhead of OLSR with *DT_MIAD* is up to 22.5% less than that of standard OLSR with small refresh interval. The overhead of OLSR with *DT_MIAD* increases as the nodes move faster.

To summarize, *DT_MIAD* algorithm satisfies both of the requirements described in section III-A with bounded throughputs and overheads. The simulation results show that, *DT_MIAD* outperforms the standard proactive routing algorithm in terms of the balance of throughput and overhead.

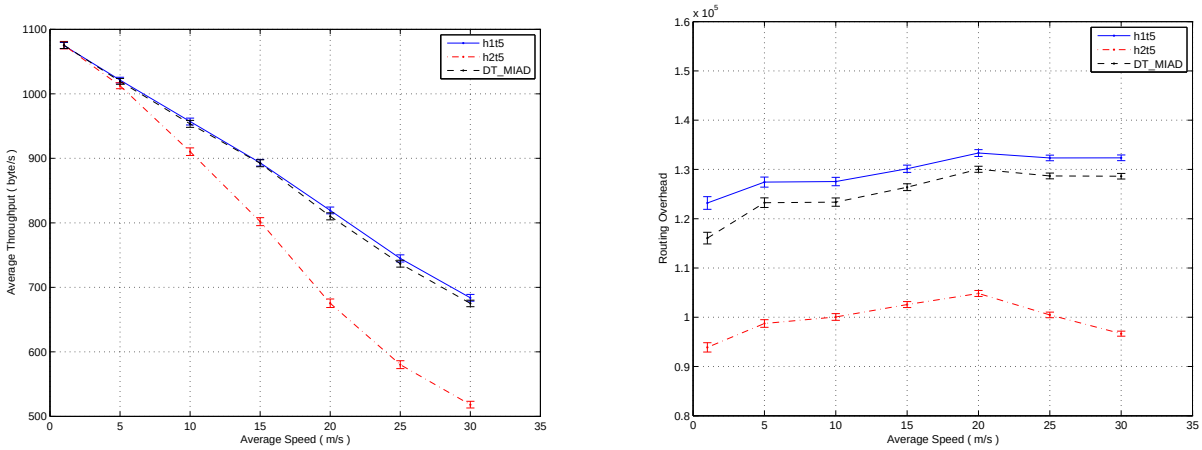
B. Routing Performance under DT_ODPU

From Fig 8(a) and Fig 6(a), OLSR with *DT_ODPU* performs slightly worse than OLSR with *DT_MIAD*, since in some cases the throughput of OLSR with *DT_ODPU* is



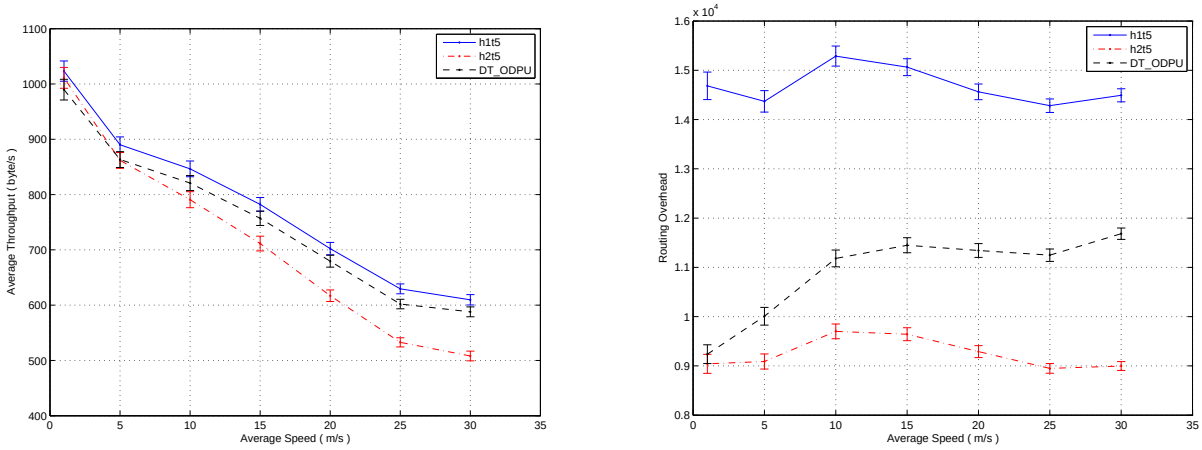
(a) Throughput (b) Overhead

Fig. 6. Performance of *DT_MIAD* (n=20)



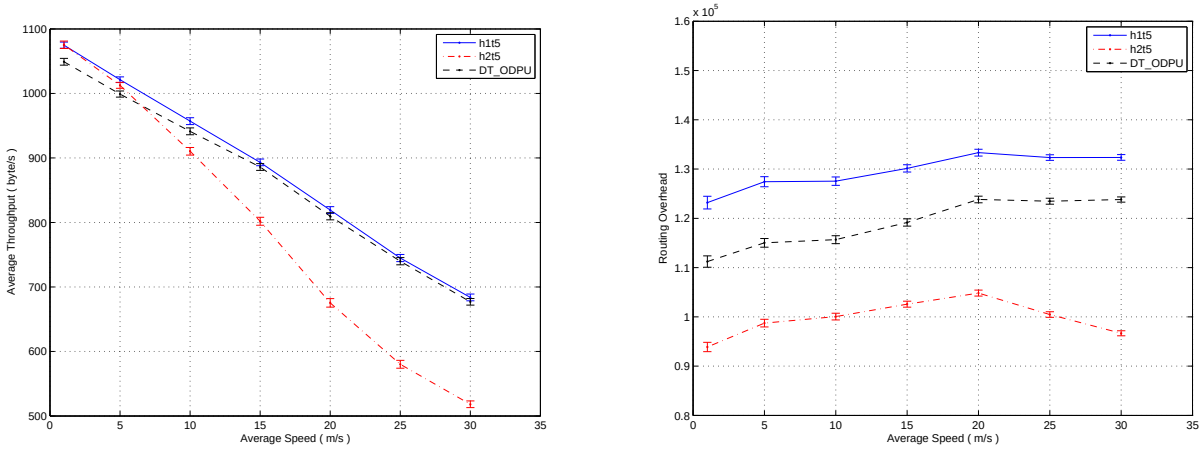
(a) Throughput (b) Overhead

Fig. 7. Performance of *DT_MIAD* (n=50)



(a) Throughput (b) Overhead

Fig. 8. Performance of *DT_ODPU* (n=20)



(a) Throughput (b) Overhead

Fig. 9. Performance of *DT_ODPU* (n=50)

significantly lower than standard OLSR with smaller refresh intervals. However, in terms of control overhead, as shown in Fig 8(b) and Fig 9(b), OLSR with *DT_ODPU* shows better adaptability to node mobility. For example, when node velocity is relatively low, the control overhead introduced by OLSR with *DT_ODPU* is as low as that by standard OLSR with larger refresh intervals ($h = 2s$). The overhead increases with the node mobility, which indicates the refresh intervals are being tuned in response to the changing network conditions.

To summarize, compared with standard proactive routing algorithms, *DT_ODPU* significantly improves the routing performance, while introducing much smaller control overhead than traditional method (i.e. improving throughput by reducing refresh intervals).

VII. Related Work

In order to meet the need for fast mobility in Mobile Ad-hoc Networks, Benzaid et al[6] presented an FAST-OLSR extension to the Optimized Link State Routing protocol (OLSR)[1]. A fast moving node refreshes the links to its MPR nodes at a higher frequency than its non-MPR neighbors by means of Fast-Hellos. Fast-Hello messages only contain the address of its MPRs. Fast-OLSR extension aims at reducing packet loss rate while keeping the overhead reasonable.

Ramasubramanian et al[7] proposed a hybrid routing algorithm which adopts optimal routing strategies, both proactive and reactive, based on separate application-level control requirements(i.e. minimizing packet overhead, controlling delay jitter and bounding loss rate). It does this by defining proactive zones around some nodes. The nodes at a distance less than or equal to the zone radius are within the proactive zone and maintain routes proactively only to the central node. All nodes not in the proactive zone of a given destination use reactive routing algorithm to discover routes to that node. Correspondingly, adjusting the zone radius changes the extent of proactive routing and reactive routing, and the overall routing performance is affected.

Boppana et al[8] proposed an adaptive Distance Vector routing algorithm. Like DSDV[3], ADV exchanges route updates between the neighboring nodes. However, only the route entries of active nodes are advertised, which reduces the size of route update messages. In addition, route updates are triggered only under certain conditions, such as route unavailability. Trigger thresholds are used to determine whether a ”partial update” or a ”full update” is advertised.

VIII. Conclusions

In this study, we present an adaptive scheme for proactive routing protocols and propose two adaptive routing algorithms, namely *DT_MIAD* and *DT_ODPU*. We evaluate the performance of these two routing algorithms through extensive ns2 simulations over a wide range of network scenarios. The results show that the proposed dynamic timer algorithms have better adaptability and routing performance than standard proactive routing algorithms.

The proposed algorithm can be improved in several instances. Currently, *DT_MIAD* and *DT_ODPU* only react to traffic loss. In order to achieve better adaptability to network load, the soft state intervals can be adjusted by monitoring the queue length. When the queue is approaching full, for example 90% as the threshold, the soft state intervals are increased to reduce the control overhead and reduce channel overhead. Such a method is currently being implemented and the results will appear in our ongoing work.

The original data, the source codes and the scripts used in this study are all available from the authors' websites (www.cs.ucl.ac.uk/staff/y.huang/dt).

References

[1] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, a. Qayyum, and L. Viennot, “Optimized link state routing protocol,” in *IEEE INMIC Pakistan*, 2001, best paper award.

[2] B. Bellur and R. G. Ogier, “A reliable, efficient topology broadcast protocol for dynamic networks,” in *IEEE Infocomm*. IEEE, Mar. 1999, pp. 178–186.

[3] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,” in *SIGCOMM ’94: Proceedings of the conference on Communications architectures, protocols and applications*. New York, NY, USA: ACM Press, 1994, pp. 234–244.
[4] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, “Ad hoc on-demand distance vector (aodv) routing,” IETF, Request for Comments 3561, July 2003.
[5] D. B. Johnson and D. A. Maltz, “The dynamic source routing protocol for mobile ad hoc networks, Internet Draft (draft-ietf-manet-dsr-06.txt),” November 2001, work in Progress.
[6] M. Benzaid, P. Minet, and K. Agha, “Integrating fast mobility in the olsr routing protocol,” 2002.
[7] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, “Sharp: A hybrid adaptive routing protocol for mobile ad hoc networks,” in *MobiHoc*, Anapolis, MD, 2003.
[8] R. V. Boppana and S. Konduru, “An adaptive distance vector routing algorithm for mobile, ad hoc networks,” in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and communications Societies*, 2001, pp. 1753–1762.
[9] Y. Huang, S. Bhatti, and D. Parker, “Tuning olsr,” in *The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2006.
[10] Y. Huang, S. Bhatti, and S. Sorensen, “A comparison of temporal and topological soft state updates for a proactive manet routing protocol,” in *Proc. London Communications Symposium 2006*, 2006.
[11] P. Samar and S. B. Wicker, “On the behavior of communication links of a node in a multi-hop mobile environment,” in *MobiHoc ’04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM Press, 2004, pp. 145–156.
[12] “Ns2 website,” http://www.isi.edu/nsnam/ns/.
[13] “The rice monarch project: Wireless and mobility extensions to ns-2,” http://www.monarch.cs.rice.edu/cmu-ns.html.
[14] J. Y. L. Boudec and M. Vojnovic, “Perfect simulation and station-arity of a class of mobility models,” in *Proc. IEEE Infocom Conference*, Miami, USA, 2005.
[15] S. Corson and J. Macker, “Mobile ad hoc networking (manet): Routing protocol performance issues and evauluation considerations,” IETF, Request for Comments 2501, January 1999.