

RICA: A Receiver-Initiated Approach for Channel-Adaptive On-Demand Routing in Ad Hoc Mobile Computing Networks

Xiao-Hui Lin, Yu-Kwong Kwok, and Vincent K. N. Lau

Department of Electrical and Electronic Engineering, The University of Hong Kong

Abstract -- To support truly peer-to-peer applications in ad hoc wireless mobile computing networks, a judicious and efficient ad hoc routing protocol is needed. Much research has been done on designing ad hoc routing protocols and some well known protocols are also being implemented in practical situations. However, one major drawback in existing state-of-the-art protocols, such as the AODV (ad hoc on demand distance vector) routing protocol, is that the time-varying nature of the wireless channels among the mobile terminals is ignored, let alone exploited. This can be a severe design shortcoming because the varying channel quality can lead to very poor overall route quality, in turn result in low data throughput. In this paper, by using a previously proposed adaptive channel coding and modulation scheme which allows a mobile terminal to dynamically adjust the data throughput via changing the amount of error protection incorporated, we devise a new receiver-initiated algorithm for ad hoc routing that dynamically changes the routes according to the channel conditions. Extensive simulation results indicate that our proposed protocol are more efficient in that shorter delays and higher rates are achieved.

Keywords: mobile computing, ad hoc networks, routing protocols, on-demand routing, channel state dependent, adaptive.

I. INTRODUCTION[†]

To realize efficient information exchange in a peer-to-peer manner in an ad hoc wireless mobile computing network (e.g., a network consisting of personal digital assistants, notebook computers, and cell phones is formed in an ad hoc manner to perform file swapping), a judicious routing protocol is needed for the source to locate the destination in the network [8]. There are two major classes of ad hoc routing protocols: on-demand and table based. As many researchers have pointed out [8], table based algorithms are notoriously inefficient in that they require periodic update of the routing information stored in the routing tables, even when there is no data traffic. The major merit of table based algorithms, as compared with on-demand algorithms, is that the set up delay for a data transfer is expected to be shorter because a route is presumably stored in the table for use. However, such route may no longer exist or usable when the actual data transfer is to be taken place for at least two reasons. First, due to the mobility of the mobile terminals in the network, their geographical locations may have changed when a data transfer is required, rendering a previously set up route useless. The second reason, which, we believe, is a more important one, is that the quality of the channels among the mobile terminals is inevitably time-varying (due to shadowing and fast fading) [7], and thus, the links in a route may no longer be usable even if the geographical loca-

tions do not change much. Indeed, this is a major consideration overlooked in previous researches on ad hoc routing protocols [1], [2], [6], [8], [9], [10], [12].

In our study, we consider on-demand routing algorithms for ad hoc networks. In particular, we are interested in studying the behavior and performance of routing protocols when the time-varying nature of wireless channels is taken into account. Recently, we have proposed a reactive routing algorithm, called BGCA (bandwidth-guarded channel adaptive) protocol [13] that is based on similar principles as in the ABR (associativity based routing) protocol [6], [10], [12]. In this paper, we propose a new receiver-initiated ad hoc routing protocol, called the RICA (receiver-initiated channel adaptive) protocol, which employs similar strategies as the AODV (ad hoc on-demand distance vector routing) [8], [9], [10] protocol.

Both the BGCA and RICA protocols work by adaptively changing the routes according to the current channel conditions. In BGCA, normally the change of a route (not a broken one) is due to the deterioration of links' quality in the route, so the intermediate mobile terminal must find a partial route to substitute the original one. This algorithm is a little "passive or reactive" to the change of the route. That is, only when the channel quality of the link drops below the bandwidth requirement of the traffics does it take actions to find a new route. In RICA, the source periodically receiving CSI (channel state information) checking packets from the destination, it can then select the shortest route based on these CSI packets even when the original route is still good, so this algorithm is an "aggressive or proactive" one compared with BGCA. In RICA, the source can adapt to the CSI change more timely than BGCA does. But the price to paid is that the amount of routing overhead is greater due to the periodical broadcast CSI checking packets. In our study, we also study the relative strengths and weaknesses of these two new protocols.

This paper is organized as follows. In Section II, we describe the RICA protocol in detail with illustrative examples. Due to space limitations, we do not provide a detailed description of the BGCA, ABR, AODV, and link state [8] protocols that are considered in our study. For an excellent survey on ad hoc routing protocols, the reader is referred to [10]. Section III contains the performance results of a quantitative comparison of the five algorithms. We provide some concluding remarks in Section IV.

II. RECEIVER INITIATED CHANNEL ADAPTIVE ROUTING

Before describing the proposed protocol in detail, we introduce the channel model below. Throughout the paper, we assume a multi-code CDMA MAC (multiple access control) layer [4] is used in all the protocols.

[†] This research is supported by HKU URC seed grants under contract numbers 10203010 and 10203413, and by a RGC research grant under contract number HKU7024/00E.

A. Channel Model

The wireless channel between every two mobile terminals is time-varying and hence, the throughput of the channel is also a time-varying function. Specifically, using a channel adaptive coder and modulator proposed previously (called ABICM) [5], the transmitter/receiver can dynamically adjust the level of error protection in the data transmission according to the channel state (when the channel state is good, less protection is included and vice versa) and, as such, the effective throughput of the channel is dynamically changing according to the channel conditions. For the details of the ABICM scheme and its applications in MAC protocols, the reader is referred to [5].

We model the channel by incorporating the fast fading and long term shadowing effects [7]. Based on the CSI (channel state information) of the channel, we divide the channel quality into 4 classes: A, B, C, and D, with a throughput of 250 kbps, 150 kbps, 75 kbps, and 50 kbps, respectively (after adaptive channel coding and modulation; see [5] for details). We define a CSI-based “hop” in the following manner: if a link between two terminals with channel quality of class A (with the throughput of 250 kbps), then the distance between two terminal is defined as ONE hop. We use this “distance” as a baseline. Then, if a link between two terminals with a channel quality of class B (with a throughput 150 kbps), the distance between two terminals is 1.67 hops because the transmission delay now is 1.67 times compared with a link of class A. Thus, for a link is with a throughput of 250 kbps, 150 kbps, 75 kbps, and 50 kbps, the distance are 1, 1.67, 3.33, and 5 hops, respectively.

RICA is a channel adaptive routing algorithm. The major feature of RICA is to make use of this channel-varying property and let the routing between source and destination adapt to the CSI of the *whole route*, or we call this adaptive-routing based on CSI. This means that in RICA the entire route between the source and destination terminals is changing with time, instead of just changing a few links, as in the BGCA algorithm.

B. Route Discovery

In the proposed RICA protocol, the source does not keep a route to any possible destination unless it is necessary (when it has packets to send to that destination). When the source terminal has packets to transmit, it must first find a route to the destination. The source terminal generates a RREQ (route request) packet which includes the following information: type of the packet, source address, destination address, hop-count from the source, and broadcast ID of RREQ. Each time the source generates a RREQ, the broadcast ID increases by one. The source and destination addresses together with the broadcast ID uniquely identify a RREQ.

After the source generates this RREQ, it sets the hop-count field to zero and broadcasts this packet out in search of the destination terminal. Any intermediate terminal receiving this RREQ first checks whether it has seen this packet before by looking up its history table, which stores the records of all the RREQ’s the terminal has received. If yes, this packet is dis-

carded; otherwise, the terminal records this packet in its history table including the following information: source and destination addresses, and the broadcast ID. The intermediate terminal must also remember its upstream terminal from which it receives the first RREQ so that it knows to which terminal it should forward the RREP (if this terminal is in the route chosen by the destination). The intermediate terminal also measures the CSI of the link through which this RREQ comes and computes the related hop distance (CSI based) from the upstream terminal. Then, the intermediate terminal resets the hop-count to original hop-count plus the hop distance to the upstream terminal. After doing all these, the terminal rebroadcasts this RREQ packets to its surrounding terminals. This process continues until the RREQ reaches the destination.

Figure 1(a) shows the broadcast of RREQ in the network. At last the destination terminal receives several RREQ’s with the same source from all possible routes. The destination also knows the hop-count (as defined earlier) distance of these routes and it chooses a route with the minimal distance value (in hop count). In Figure 1(a), a RREQ reaches the destination terminal through 3 routes (note that the links are labeled with the channel classes) with the hop count distance 6, 7, and 4.33, respectively. The destination terminal generates a RREP which includes the following information: type of the packet, source

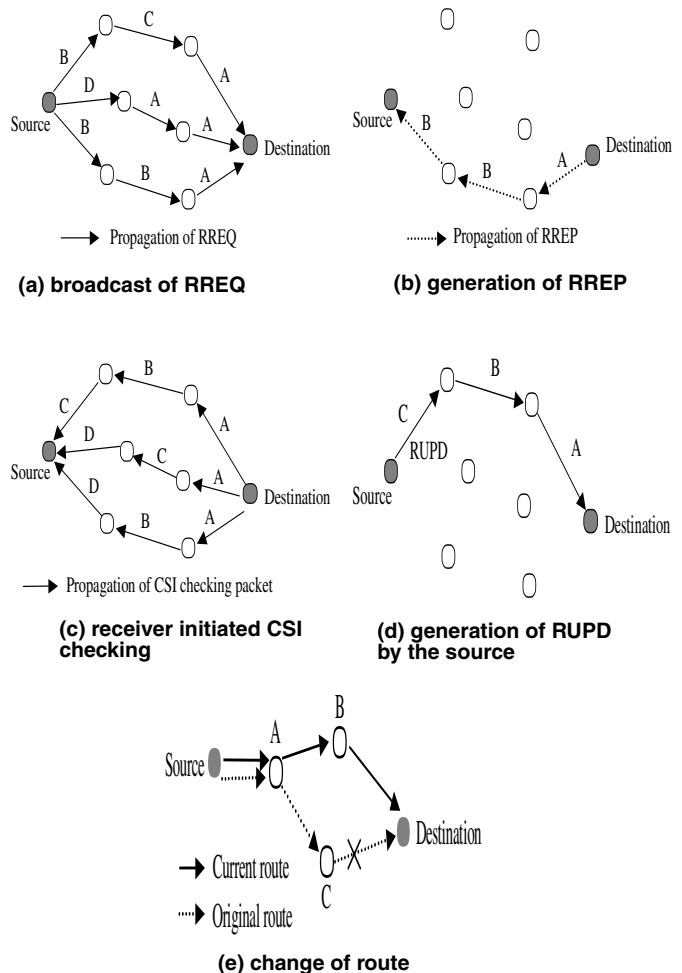


Figure 1: Illustration of the RICA protocol.

address, destination address, sequence number (corresponding to the RREQ), and hop count of the route. The destination terminal then unicasts this RREP along the route (shortest in hop count) to the source terminal (note that each terminal knows its upstream terminal to which to forward the RREP) as shown in Figure 1(b). After receiving this RREP, intermediate terminal sets its route entry to the destination as valid and update the following information: the source and destination addresses, the next upstream and downstream terminals (also begins to use the related CDMA PN (pseudo-random noise) codes [4] to receive and transmit packets). When the RREP reaches the source, the source can transmit packets to the destination terminal.

C. Broadcast of CSI-checking Packets

Because the channel quality between two terminals is a time-varying function, the throughput of the route to the destination is also changing with time, as discussed earlier. The essence of RICA is to make routing to the destination adaptive to this signal fading (due to multipath propagation [7]) environment such that a route can be kept with a high throughput to the destination even under this CSI fluctuating environment. The key idea is to let the destination terminal broadcast a CSI checking packet periodically (for example every second, this has to be decided by the change speed of the link CSI), this checking packet is used to measure CSI of every link it passes, then converts this CSI value to hop count. At last the source receives several CSI checking packets from the destination, then it can choose the shortest one as the new route.

We explain this process in more detail with the help of Figure 1(c) and Figure 1(d). First, the destination terminal generates a CSI checking packet which includes the following information: type of the packet, source and destination ID's, broadcast ID, hop count[‡] (based on CSI) from the destination, and TTL (time-to-live) field. Every time the destination broadcasts a new CSI-checking packet, the broadcast ID increases by one. The TTL field is used to limit the broadcast scope of the packet because a full broadcast is avoided to save bandwidth. The TTL field is set to the originally known hop distance (not based on CSI) of the path.

Every time the packet is rebroadcast, the TTL field is decreased by one and when TTL is zero, this packet is discarded. An intermediate terminal receiving this checking packet resets the hop count field based on CSI as mentioned before and decreases the TTL by one and rebroadcasts this checking packet out. In this rebroadcast checking packet, the intermediate terminal must specify from which terminal (may be this terminal is the possible downstream terminal) it receives this checking packet.

In the above mentioned manner, the possible downstream terminal can also overhear this packet and set the intermediate terminal as its possible upstream terminal, and knows which CDMA PN code the upstream terminal uses to send packets to it. It is now ready to receive packets from the possible

upstream terminal and continues detecting this PN code for a period of time (if this PN code now is not used to receive packets, we set this detecting period to 100ms, during this period, if no packet to the destination is transmitted using this PN code, the terminal stops detecting the PN code and set the route entry to the destination as invalid). The intermediate terminal must also remember the downstream terminal from which it receives the first checking packet (in the future it can use the corresponding PN code to send packets to this downstream terminal).

Note that a terminal only broadcasts a checking packet once, and if it receives the further copy of the same packet it discards it. At last, the source terminal receives several checking packets from all possible routes as shown in Figure 1(c), then the source terminal selects the shortest path and uses it to substitute the original route. (Three candidate routes in Figure 1(c) with hop-count of 6, 9.33, and 7.67, respectively.) The source terminal then sends a route update (RUPD) packet to its next downstream terminal as shown in Figure 1(d), then this downstream terminal is also ready to receive packets and set the route entry as valid. Up to now, the new route is set up and can be used to transmit packets. The first transmitted packet has an update field. On receiving this first packet, the downstream terminal updates the route entry including the following fields: source and destination IDs, upstream and downstream terminals, and sets the downstream terminal from which receives the first CSI checking packet as the next hop to the destination. Note that:

- the original route at last automatically expires (because no packets have been transmitted through it during specified timeout period, for example 1 second) and be deleted; and
- the break of the link in original route has no influence to the data transmission in the current route if this link is not in the current route, this will be illustrated in detail below.

D. Route Maintenance

In RICA, the update of the route entry can be frequent, so an upstream terminal A must be sensitive to the connection with its downstream terminal B. To ensure that, the downstream terminal B sends an ACK (acknowledgment) packet to confirm the receipt of data packet. This ACK packet is sent with another PN code (note that A send packet to B using the PN code $PN(A, B)$, while B sends packet to A using PN code $PN(B, A)$, these two codes are different). When a terminal notifies that its downstream terminal has moved out of its transmission range, it generates a REER (route error) packet which includes the following information: source and destination ID's, and the terminal's own ID, then the terminal unicasts REER to the upstream terminal. The upstream terminal first checks whether the terminal unicasting the REER is its downstream terminal by looking up the related route entry. If not, it ignores this REER because this REER comes from a broken route which is out of date and has no influence on the data transmission that is going on in the current route.

For example, in Figure 1(e), terminal C finds that the link to destination is broken and send a REER to A, but A igr

[‡] The hop count is set to zero at the beginning.

REER because it knows that terminal C is not its downstream terminal and REER comes from an old link that is not used by the current route. If the terminal unicasting the REER is its downstream terminal, it also unicasts this REER to its upstream terminal. The process continues, and if the REER reaches the source, then source can decide whether it should initiate another RREQ based on two situations: 1) source terminal now is receiving CSI checking packets, then the source terminal ignores the REER and chooses the shortest route based on CSI checking packet; 2) if source terminal is not receiving CSI checking packets, it broadcasts a RREQ in search of the destination and waits for a RREP, there are three scenarios:

- if the RREP reaches the source together with the CSI checking packets (the source terminal waits 40 ms so that it may receive all the CSI checking packets, during this period, RREP also reaches the source), the source selects the shortest route based on CSI checking packets and RREP packet;
- if the CSI checking packets arrive before the RREP, the source decides the route based on these CSI checking packets; afterwards, if RREP also arrives, the source chooses the route based on RREP;
- if the RREP arrives before CSI checking packet, the source chooses route based on RREP, afterwards, the CSI checking packets arrive, the route is decided based on CSI checking packets.

III. RESULTS

In this section, we present the results obtained in our extensive simulations comparing the five algorithms (RICA, BGCA, ABR, AODV, and link state) considered in this paper. We first introduce the simulation environment.

A. Simulation Environment

The simulation parameters we used are as follows:

- number of terminals: 50;
- testing field: 1000m × 1000m ;
- mobile speed: uniformly distributed between 0 and MAXSPEED (will be elaborated later);
- mobility model: when the terminal reaches its destination, it pauses for 3 seconds, then randomly chooses another destination point;
- radio transmission range: 250 m;
- channel model: capturing the fast fading and long term shading factors, there are four different channel conditions with throughput 250 kbps, 150 kbps, 75 kbps, 50 kbps respectively;
- bandwidth of the common channel: 250 kbps, we suppose this channel is robust that can withstand deep fading and interference;
- MAC of common channel: unslotted CSMA/CA based on CDMA [4];
- traffic load: 10 terminal pairs, in each pair, we change the traffic load for 10 and 20 packets/sec, respectively.

Furthermore, the size of the data packet is 512 bytes and the

capacity of data buffer size is set to be 10 packets for one connection of two adjacent mobile terminals. This is because we would like to have a fair comparison of the protocols under different load and we do not want the buffer size to become the bottleneck of the protocol performance when the load is extremely high. Furthermore, we do not set the buffer size to a great value, and this means that when the link is in deep fading, the packets cannot be sent out timely, congestion results and lead to the drop of the packets. The aim of such experiment is to test all the algorithms under the same channel fading wireless environment.

The transmission of packet is a store-and-forward process. When packet reaches an intermediate terminal, it waits in the queue for service (FCFS). Each packet is allowed to be kept in the buffer for no more than three seconds, and if it has not been transmitted in this period, it is discarded. The generation of data packets in each source terminal follows a Poisson arrival process, i.e., the inter-arrival of two packets is exponential distributed.

Each simulation is run for 500 seconds (simulation time) and repeated for 25 trials. We compute the average of the results of these 25 sets of data. To evaluate the five routing algorithms, we compare them using three metrics:

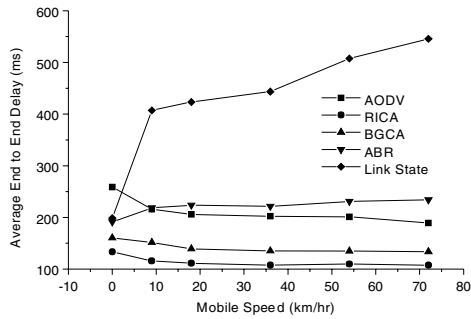
- **Average End-to-End Delay:** Measured in ms, the end-to-end delay includes the processing time and queuing time of packet in each terminal in the route.
- **Successful Percentage of Packet Delivery:** This is the ratio of packets reaching the destination to total packets generated in the sources. A packet may be dropped if there is not enough data buffer due to the congestion, or has stayed in the buffer for more than three seconds.
- **Routing Overheads:** This is measured in bps. We count the total routing packets and data acknowledgment packets in each round of simulation. Each time the common channel is used to transmit a routing packet, this is counted as one transmission. We average the amount of routing overheads (in bits) to the whole simulation time.

For the link state protocol, at the beginning of each simulation run, an accurate view of the network topology is installed in each mobile terminal. When the mobile terminal finds the bandwidth with its neighbor changes (due to CSI change or link break), it floods this change throughout the network. The aim is to test the performance of the protocol and see whether it can converge or adapt to this time-varying wireless environment.

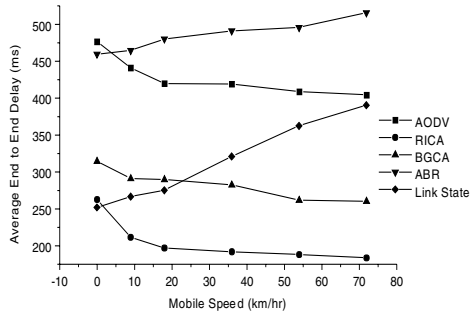
B. Average End-to-End Delay

The first set of results is the average end-to-end delay against the mobile speed for the traffic load of 10 packets/sec and 20 packets/sec. We varied the mean mobile speeds from 0 to 72 km/hr, and thus, the value of MAXSPEED was varied from 0 to 144 km/hr. As can be seen in Figure 2, taking the CSI into consideration can greatly shorten the transmission delay from the source to destination as in BGCA and RICA. RICA outperforms the other three algorithms for the following reasons.

1. The source can update the route to the destination



(a) 10 packets per second



(b) 20 packets per second

Figure 2: Average end-to-end delays of all protocols.

quently and usually this route is temporally the shortest one, and this greatly speeds up the transmission of the data packets.

2. The periodical update of the route is adaptive to the regional changes of the CSI, and this happens very frequently, and thus, in this sense, the packets may reach the destination through different routes. This means that load balancing can be achieved in RICA and each link is not overloaded for a long time, thus the queuing length is decreased and transmission delay is also shortened.
3. CSI checking packets sometimes make the full broadcast in search of a route unnecessary, this reduces the data queuing delay at the source because source terminal can choose a route to destination based on CSI checking packets. In BGCA, however, route update is not so frequent as in RICA because BGCA is not so sensitive to the CSI change of the route, BGCA cannot ensure the route to the destination is the shortest all the time. BGCA takes action (performs local query for a partial route) only when a link is in deep fading and may cause congestion. But this routing algorithm also takes the CSI into consideration, and ensures that the throughput of the link can satisfy the bandwidth requirement of the traffic in most of the time. This also greatly reduces the transmission time.

In BGCA and RICA, the delay decreases with the increase of the mobile terminal. This is because when the mobile speed increases, the long queue is not easy to form (because link break happens more often), thus decreasing the queuing delay, but at the same time the number of dropped packets also increases, as detailed below. In ABR, however, delay increases with the mobile speed because of the local search. When the

link breaks, the packets accumulate in the upstream terminal performing the local search until a partial route is found, thus the long queue forms and queuing time increases.

We also observe one interesting phenomenon that when in low mobility ABR outperforms AODV, but in high mobility AODV outperforms ABR in end-to-end delay. This is because ABR takes the load and propagation delay of the link into consideration when selecting the route (by not choosing links with heavy load), thus balancing the link load and decreasing the delay. While in AODV, the destination responds only the first RREQ and chooses the path this RREQ has gone through although this route is usually not the shortest one or some links in the route may be congested. But as the mobility increases and the link is easier to break due to the mobility, in AODV, the source terminal performs a full broadcast in search of a new route, and packets in the original broken route usually is discarded, so the long queue will not be easy to form.

However, in ABR, a LQ (local query [8]) is implemented to find a partial route and data packets have to wait in the terminal performing LQ, so the long queue is formed, and this increases the end-to-end delay, but on another side, the packet delivery rate of ABR is also greater than that in AODV as shown later (usually in AODV a great portion of data packets is dropped due to link break as observed in our experiments). Another reason is that usually, the link in ABR is robust than that in AODV, so the long queue is easier to form in the link with low throughput (50 kbps or 75 kbps). Long queue is also formed in link with low throughput in AODV, but frequent link breaking often eliminates these long queues. Normally, route in ABR is longer than that in AODV (as will be seen in Section E) because of the different route selection criteria, this also makes the delay in ABR longer.

From Figure 2, it can be seen that, the end-to-end delay in link state protocol increases more sharply with the mobility due to the formation of routing loop [3]. In the link state protocol, the change of the link is broadcast throughout the network. This idea is very effective in the wire-line network where the link cost is relatively stable and the algorithm can quickly converge. However, in an ad hoc wireless network, this is not the case because the CSI or network topology changes too frequently, and each change has to be flooded as routing packet throughout the network through the common channel. This flooding leads to an inefficient use of the common channel and the frequent collisions of the packets. The consequence is that at last the status of the network in each mobile terminal can be very inconsistent and thus, the link state algorithm takes a long time to converge.

Furthermore, as the mobile terminal's mobility increases, the link breaking event happens more frequently but this information cannot be propagated timely throughout the network due to collisions (we observe in the simulations that the common channel is very congested for the link state protocol), and thus, a routing loop is formed. The routing loop, in turn, causes:

- the increase of the packet delay; and
- the severe contention of the data buffer and eventually the drop of packets.

We observe that when the network is static or in low mobility, the delay of the packet is very low (even the lowest in some scenarios). This is due to two reasons:

1. At the beginning of each simulation, a correct view of the network has been installed in each mobile terminal and the network topology is relatively stable during the simulation time due to the low mobility. This may be “unfair” to other protocols.
2. Usually the bandwidth of the links that the packet has gone through are very high due to the property of Dijkstra algorithm’s (used in the link state approach) route selection criteria (as further elaborated in Section E).

We also observe an interesting phenomenon in the link state protocol that as the traffic load increased (from 10 packets/sec to 20 packets/sec), average end-to-end delay of the packets decrease (when the mobile terminals are in motion). This is utterly different from other routing protocols. The reason is simple: as the mobile terminal is in motion, the routing loops are formed in the network. Usually a loop lasts for several seconds from our observations, and thus, when we increase the traffic load, congestion is much more easier to form in the loop because of the limited data buffer size (10 for these three scenarios). Consequently, the packets in the loop are dropped with a much higher probability and the average time a packet staying in the loop decreases. Eventually, those packets reaching the destination are from a loop-free route or a route with loop with a short life-time. This interesting phenomenon also reinforces our conclusion that the mobility of mobile terminals is the main cause of the formation of the loop (note that as the network is static, the delay in link state protocol increases with the increase of the traffic load as in other protocols).

C. Successful Percentage of Packet Delivery

From the simulation results shown in Figure 3, we can see that taking CSI into consideration also contributes to the reliability of packet delivery. Again in terms of successful delivery percentages, RICA outperforms the other 4 algorithms for the following reasons.

1. Usually links in RICA are with high throughput (see Section E) and this ensures that data packets will not be discarded due to the long queue (not enough buffer).
2. Frequent and adaptive update of the route can make the traffic evenly distributed in the network, thus no link is unfairly overburdened and link congestion does not happen.
3. Packets do not accumulate in a particular terminal because of load balancing, thus long queue seldom forms in a link and the drop of large amount of queuing packets because link break seldom happens.

The gain is more obvious as we increase the traffic loads. In BGCA, the update of route does not happen so often as in RICA. The route update in BGCA only happens when it is broken or in deep fading and the source has to find a new partial route to substitute it, so the packet queues in BGCA are longer than those in RICA and when the link breaks, data loss is more serious. But in BGCA, remedy (i.e., LQ) is taken to ensure the

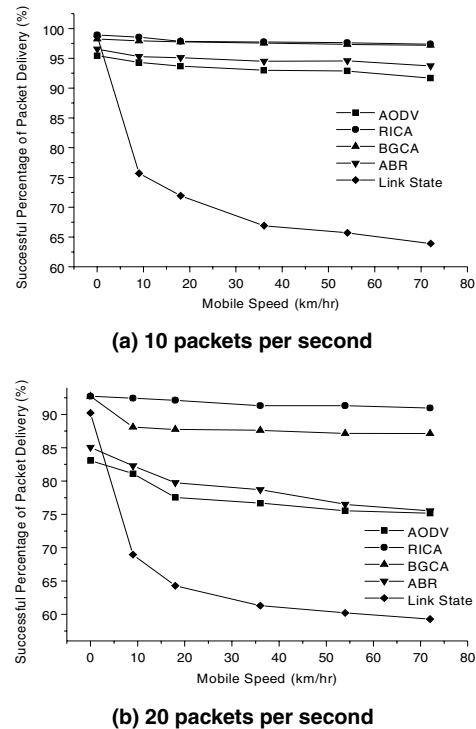


Figure 3: Successful percentage of packet delivery of all protocols.

bandwidth requirement be satisfied, so the congestion is avoided and when link breaks, data loss is reduced. ABR and AODV do not take the CSI into consideration, so their routing cannot adapt to the change of link throughput which fluctuates with time and long queue is easier to form.

Normally the main causes of data loss are: link congestion and not enough data buffer; and link break. In these two algorithms, long queue is very easy to form in the link with low throughput especially when the traffic load is high (for example 20 packets/sec). We have observed the saturation of the data buffers in this circumstance for many times in our experiments. To ensure the reliability of packet delivery, long queue should be avoided. As seen from the results of delivery rate, ABR performs better than AODV because:

- the routes in ABR are more robust than those in AODV;
- ABR takes the link load into consideration when choosing the route; and
- ABR performs LQ to find a partial route at the broken point, so the probability of packets being dropped in the upstream route is reduced.

As expected, packet delivery rate decreases with increase of the mobility and traffic loads in 4 on-demand routing protocols because the link break happens more often and congestion and long queue are easier to form.

In the link state routing protocol, the packet delivery rate drops more sharply with the increase of the mobile speed due to the formation of the routing loop. The higher the mobile speed, the easier for a routing loop to form (see Section E). This illustrates that link state protocol is not suitable

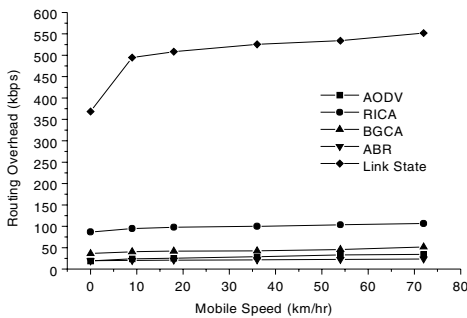
mobile wireless ad hoc network. The common channel is over-used and the routing packets cannot be propagated efficiently throughout the network.

D. Routing Overhead

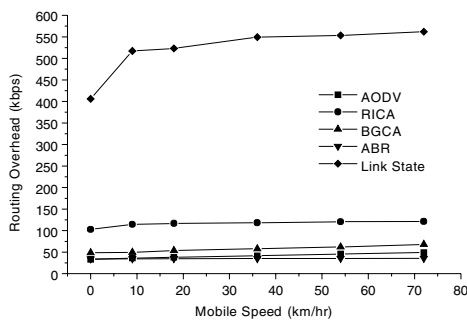
Routing overhead is defined as the average bit rate required for sending/receiving the routing messages. The results on routing overhead are shown in Figure 4. Taking CSI into consideration when choosing a route can improve the network performance in the sense of delay and packet delivery rate, but the cost is that it also adds more routing overhead. Using the amount of routing overhead in AODV as a baseline, BGCA and RICA generate about 1.5 and 4 times more overhead, respectively. The reason is obvious: in RICA, the destination broadcasts a CSI checking packet periodically to the source so that the source can master the CSI changes timely and change the route adaptively; in BGCA, in order to ensure the bandwidth requirement of traffic is satisfied, the intermediate terminals have to perform local search, this also increases the routing overhead. As seen from the plots, ABR generates the least of routing overhead because:

- the route in ABR is long-lived so the break of link happens not so frequently as in other routing algorithm; and
- even when the link breaks, the intermediate terminal performs local search instead of a full broadcast.

Thus, ABR is a bandwidth efficient algorithm. On the contrary, the amount of routing overhead in link state protocol is much higher than in other protocols. In link state protocol, each change of the link cost is broadcast throughout the network even though much routing information is useless. This causes a



(a) 10 packets per second



(b) 20 packets per second

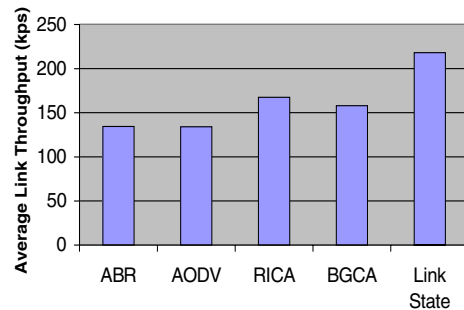
Figure 4: Routing overhead (the average bit rate of routing messages) of all protocols.

tremendous amount of routing overhead. This inefficient use of channel causes congestion and can increase the consumption of the limited battery power in each mobile terminal [11], [14].

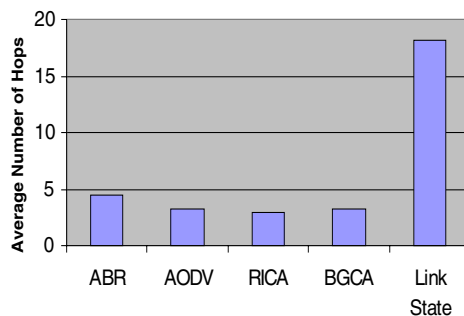
In summary, in all algorithms, routing overhead increases with the mobility because link break is more frequent and this increases the load of route maintenance. It is also observed that increasing the load of data traffic only has little influence on the routing overhead because more data acknowledgments are generated.

E. Quality of Routes

It is also interesting to compare the quality of the routes selected by different algorithms. Figure 5(a) shows the *average link throughput*, which is defined as the total bandwidth of the links that all packets reaching destinations have passed through, divided by the total number of hops that these packets have passed through. This parameter reflects the quality of the selected routes in each routing algorithm. As can be seen, the link throughputs in ABR and AODV are very close to each other and are the lowest among all protocols because these two algorithms have not taken the CSI of the link into consideration when choosing a route. In RICA and BGCA, the average link throughputs are much higher than those in ABR and AODV because the former two are adaptive to the CSI change of the link when routing packets. This is the major reason of the algorithms' ability in reducing the packet delay. We can also see that the link throughput in BGCA is a little lower than that of RICA because BGCA is a little "passive" or "conservative." That is, only when a link is in deep fading and cannot satisfy the bandwidth requirement does it take measure to find a par-



(a) average link throughput



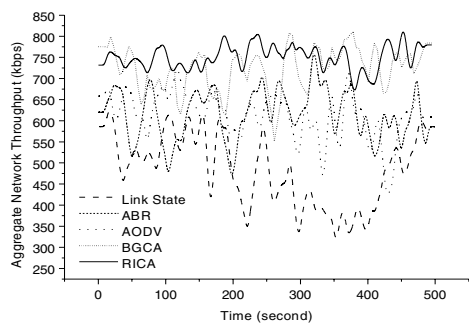
(b) average number of hops

Figure 5: Comparison of route quality of all protocols.

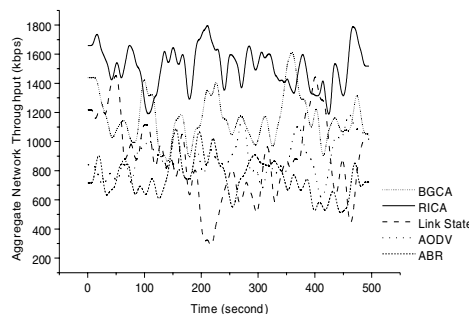
tial route to substitute the original one. Link state routing protocol has the highest average link throughput due to the route selection criteria of the Dijkstra algorithm (when a mobile terminal need to forward packets, it uses this algorithm to compute the next hop, normally the link throughput between the mobile terminal and next hop is very high, for example 250 or 150 kbps). This observation may seem to be contradictory to the results of link state protocol but in fact, there is another counteractive factor, as detailed below.

Figure 5(b) shows the average number of hops of the route in each algorithm. The testing mobile speed is 72 km/hr for each algorithm. As can be seen, the route in link state protocol has the highest number of hops due to the formation of the routing loop. This leads to a severe deterioration of the performance even it has the highest average link throughput. That is, even the throughput is high, the queuing delay due to route loops is so high that it offsets the gain from a higher throughput. The route in RICA has the lowest number of hops because this algorithm can continuously find the shortest route. The length of the route in ABR are longer than the other three on-demand routing protocols because ABR inclines to select the route with the highest stability and normally such a route has a greater number of hops.

Finally, as shown in Figure 6, we also measured *overall throughput*, which is defined as the amount of data reaching destination terminals in every 4 seconds (simulation time). As can be seen, BGCA and RICA consistently outperform the other protocols in this aspect.



(a) 20 packets per second



(b) 60 packets per second

Figure 6: Throughput variations of all protocols.

IV. CONCLUDING REMARKS

In this paper, we study the behavior and performance of ad hoc routing protocols under a more realistic channel model. We also propose a new channel-adaptive routing protocol, called RICA (Receiver Initiated Channel Adaptive) protocol, which takes into account the time-varying nature of the channel and incorporate an adaptive channel coding and modulation scheme for dynamically adjusting the throughput (the amount of error protection) according to the channel conditions. In our extensive simulations study, we found that both the RICA and our previously proposed BGCA protocols outperform the well-known ABR and AODV protocols. Furthermore, the performance of RICA is slightly better than that of BGCA, indicating that changing the entire route according to channel conditions is more efficient than just incrementally changing each link in the route.

REFERENCES

- [1] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. MOBICOM'98*, pp. 85–97, July 1998.
- [2] M. S. Corson, J. P. Macker, and G. H. Cirincione, "Internet-Based Mobile Ad Hoc Networking," *IEEE Internet Computing*, July/Aug. 1999, pp. 63–70.
- [3] J. J. Garcia-Luna-Aceves and S. Murthy, "A Path-Finding Algorithm for Loop-Free Routing," *IEEE/ACM Trans. Networking*, vol. 5, no. 1, pp. 148–160, Feb. 1997.
- [4] K. I. Kim, *Handbook of CDMA System Design, Engineering, and Optimization*, Prentice-Hall, 2000.
- [5] V. K. N. Lau, "Performance of Variable Rate Bit-Interleaved Coding for High Bandwidth Efficiency," *Proc. of VTC'2000*, vol. 3, pp. 2054–2058, May 2000.
- [6] S.-J. Lee, M. Gerla, and C.-K. Toh, "A Simulation Study of Table-driven and On-demand Routing Protocols for Mobile Ad Hoc Networks," *IEEE Network*, vol.13, no.4, pp. 48–54, July-Aug. 1999.
- [7] J. D. Parsons, *The Mobile Radio Propagation Channel*, Second Edition, Wiley, 2000.
- [8] C. E. Perkins (Ed.), *Ad Hoc Networking*, Addison-Wesley, 2000.
- [9] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing, Mobile Computing Systems and Applications," *Proceedings of WMCSA'99*, pp. 90–100, 1999.
- [10] E. M. Royer and C.-K. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, April 1999.
- [11] A. P. Sista, O. Wolfson, and Y. Huang, "Minimization of Communication Cost Through Caching in Mobile Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 4, pp. 378–390, Apr. 1998.
- [12] C.-K. Toh, "A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing," *Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, pp. 480–486, 1996.
- [13] X.-H. Lin, Y.-K. Kwok, and V. K. N. Lau, "BGCA: Bandwidth Guarded Channel Adaptive Routing for Ad Hoc Networks," *Proceedings of the 3rd IEEE Wireless Communications and Networking Conference (WCNC'2002)*, Orlando, Florida, USA, March 2002.
- [14] M. Zorzi and R. R. Rao, "Error Control and Energy Consumption in Communications for Nomadic Computing," *IEEE Transactions on Computers*, vol. 46, no. 3, pp. 279–289, March 1997.