A Routing Protocol for Utilizing Multiple Channels in Multi-Hop Wireless Networks with a Single Transceiver

Jungmin So
Dept. of Computer Science, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Email: jso1@uiuc.edu

Nitin Vaidya
Dept. of Electrical and Computer Engineering, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Email: nhv@uiuc.edu

Technical Report

October 2004

***Abstract*— Wireless ad hoc networks often suffer from rapidly degrading performance as the number of users increases in the network. One of the major reasons for this rapid degradation of performance is the fact that users are sharing a single channel. Although widely deployed standards such as IEEE 802.11 provides multiple non-overlapping channels, most protocols for wireless ad hoc networks are designed to work in a single channel environment. This is because most current devices are equipped with a single transceiver, and they can only listen to one channel at a time. This constraint complicates the protocol design, which discourages the use of multiple channels in ad hot networks.**

With multiple channels, multiple communications can simultaneously take place in a region without interfering each other, resulting in increased capacity. In this paper, we consider issues regarding multiple channels at the network layer, assuming a single channel MAC protocol such as IEEE 802.11 DCF. Also, we assume that each node in the network is equipped with a single transceiver. This set of assumptions is very practical, because most devices have a single wireless card implementing the IEEE 802.11 DCF protocol. We discuss issues in multi-channel routing, and propose a routing protocol which utilizes multiple channels with a single transceiver to improve network capacity. Finally, we evaluate the performance of the proposed protocol using simulations. The results show that the proposed protocol can improve the throughput significantly, without using additional hardware or changing the MAC protocol.

I. Introduction

Wireless ad hoc networks have increasingly gained attention in the last decade. Numerous protocols have been proposed to achieve high performance in wireless ad hoc networks. However, it is well known that ad hoc networks suffer from rapid degrading performance as the number of users increases in the network. One of the major reason is that all users in the network shares a single channel.

The IEEE 802.11 standard, which is widely deployed in current devices, provides multiple non-overlapping channels. Using multiple channels, two transmissions can take place simultaneously without interfering each other. In IEEE 802.11 infrastructure mode, neighboring access points (APs) often operate on different channels to reduce interference between cells. However, in an ad hoc mode, it is often the case that all nodes in the network share a single channel. Also, most MAC and routing protocols designed for ad hoc networks assume the network uses a single channel.

The major problem in utilizing multiple channels is that current devices are often equipped with a single transceiver. Thus, a node can only transmit or receive on one channel at a time, although nodes can switch channels in a short time (80μs) [1]. If two nodes are on different channels, they cannot communicate with each other. So they need to agree on a common channel and switch to that channel in order to communicate.

Recently, MAC and routing protocols are proposed in the context of multi-channel networks. In this paper, we focus on network layer approach to utilize multiple channels. Also, we consider the environment that each node is equipped with a single transceiver. This set of assumptions is very practical, because most devices have a single wireless card implementing a single channel MAC protocol such as IEEE 802.11 DCF. Thus a solution developed under these assumptions can be easily deployed and used with current devices.

We state the assumptions we use in this paper.

- Each node is equipped with a single transceiver. Thus, a node can only listen on one channel at a time.
- A node is capable of switching channels. The channel switching delay is less than 80μs. [1].
- IEEE 802.11 DCF is used as the MAC protocol. We do not change MAC protocol at all.
- The network layer can determine when to switch channels, and which channel to switch to.

There are several routing protocols for multi-channel networks [2], [1], but all of them require multiple transceivers at each node. The proposed protocol is the first routing protocol that utilizes multiple channels without additional hardware, or any change in the MAC protocol.

The outline for the rest of the paper is as follows. In Section II, we discuss issues regarding routing in multi-channel environment. Based on the discussion, we present our proposed protocol in Section III. In Section IV, we evaluate the perfor-

mance of our proposed protocol by simulations. In Section V, we review the relevant work in this area. Finally we conclude and discuss directions for future research in Section VI.

II. Routing with Multiple Channels

In this section, we discuss the issues in designing a routing protocol for multi-channel multi-hop networks. We investigate different approaches, and identify the benefits and problems of each approach. The routing protocol proposed in the next section is a result of this discussion.

As mentioned before, our goal is to design a routing protocol that can utilize multiple channels without modifying the MAC protocol. Also, we assume that each node has only a single transceiver, and thus capable of listening to only one channel at a time. Since traditional MAC protocols such as IEEE 802.11 DCF are designed for a single channel network, the routing protocol must perform channel assignment as well as route discovery and maintenance.

Consider the scenario illustrated in Figure 1. In this scenario, five nodes form a multi-hop ad hoc network sharing a common network ID, but they are listening on different channels. If two neighboring nodes want to communicate, they must agree on a common channel on which they will communicate.

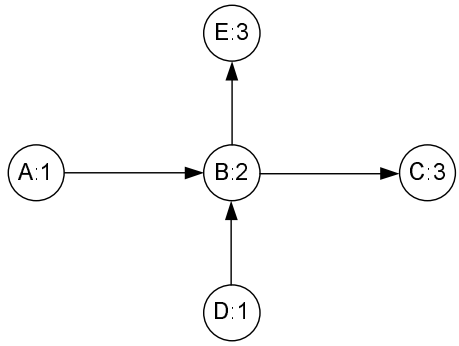

Fig. 1. An example network scenario. The label in each node indicates the node id and the channel it is listening on.

We define a *flow* to be a connection between a source-destination pair. Suppose node A wants to initiate a flow with node C. Then node A must first obtain a route to node C, if it does not already have one. For a single-channel network, establishing a route means to obtain a path where each node in the path knows the next hop towards the destination. However in a multi-channel network, each node in a path needs to know on which channel it should transmit packets, so that it can reach the next hop and eventually the destination.

There are multiple ways to assign channels, and we investigate two approaches: assigning channels to nodes, and assigning channels to flows.

A. Two approaches for channel assignment

The first approach in channel assignment is to assign channels to nodes regardless of the traffic patterns. In this approach, nodes do not switch channels to participate in a flow, but each node in the path knows the next hop node and the channel it is listening on.

Let us consider the scenario in Figure 1 again. Suppose node A is initially listening on channel 1, node B on channel 2, and node C on channel 3. For node A to send packets to C, A needs to know that B is the next hop node and it is listening on channel 2, and B needs to know that C is the next hop node and it is listening on channel 3. After establishing a route, nodes can send packets by switching to the channel of the receiver, whenever they have packets to send.

The benefit of this approach is that route establishment is decoupled with channel assignment. It makes both routing and channel assignment algorithm simple. For example, channels can be randomly assigned to nodes, and each node can broadcast its route information on all channels. With these simple mechanisms, all nodes will eventually obtain routes to every other node in the network.

However, we argue that assigning channels to nodes can result in significant performance degradation. It is due to the *deafness problem*, in which two nodes cannot communicate because they are listening on different channels. Deafness problem is first identified in the context of directional antennas [3], where nodes cannot communicate with each other because their antennas are beamforming in other directions. Suppose in Figure 1, there is a flow from A to C and another flow from D to E. Node A and D are on channel 1, node B is on channel 2, and node C and E are on channel 3. When B receives a packet from A, it switches to channel 3 to forward the packet to C. When it sends the packet, it returns to channel 2. While node B is on channel 3, node D has a packet to send to B, so it transmits the packet on channel 2. But since B cannot receive the packet, the packet will be lost.

In this case, MAC protocols such as IEEE 802.11 DCF will wait for a random delay and retransmit the packet. This results in unnecessarily increased delay. Moreover, since D does not know when B is returning to channel 2, the retransmission may fail again. After several retransmissions, DCF drops the packet and notify the routing protocol that the link has broken. Then the routing protocol thinks that the route has been broken, and take actions accordingly. The impact of this problem becomes even more severe if there is another node trying to send a packet to node D on channel 1. A series of nodes waiting on the receiver's channel may result in a channel deadlock, as identified in [4].

The second approach in channel assignment is to assign channels to flows instead of nodes. This means that whenever a route is established, all nodes in the route are assigned with a common channel. In this case, the channel assignment needs to be coupled with route establishment.

This approach works well with on-demand routing protocols, since channels are assigned at the time of route discovery. But it is difficult to maintain routes proactively. Note that for the first approach where channels are assigned to nodes, both proactive and reactive routing schemes can work.

The benefit of this approach is that once the route has been established, nodes in the path do not need to know which channel the next hop is listening on, nor they need to switch channels to transmit packets to the receiver. The deafness problem can be avoided, because node are not switching channels. The downside of this approach is that the routing protocol becomes

complicated, as explained in the next subsection. We choose this approach in this paper, but we do not claim that any multi-channel routing protocol should take this approach.

B. Assigning channels to flows

In a very dense network, it might be possible to find node-disjoint routes for all flows, but for a mid-sized network with many flows, intersecting routes can be a common case. It is not easy to assign channels if multiple flows share a same node as its intermediate or end node. For example, consider the scenario in Figure 2. Suppose there are two flows, one from A to E and another from F to I. These two flows intersect at node C. To assign a common channel to all nodes in a flow, the same channel has to be assigned to these two flows, and the benefit of having multiple channels is nullified. This problem becomes worse as the number of flows increases, because the chance of two flows intersecting at a node increases. One example of this situation is illustrated in Figure 3. Also, suppose a new route must intersect with two existing routes that are node-disjoint and operating on different channels. This route cannot be established without forcing the already existing flow to switch channels. It is certainly undesirable if the routes cannot be established even though there is a path between the source-destination pair.

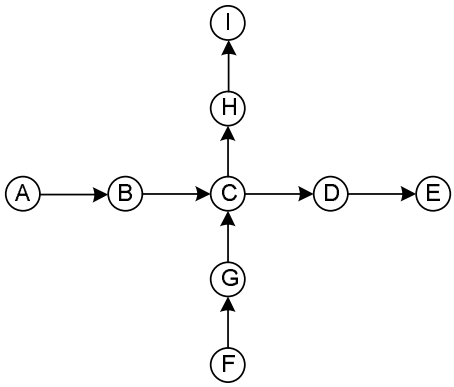

Fig. 2. An example network scenario. Two flows A-E and F-I are intersecting at node C.

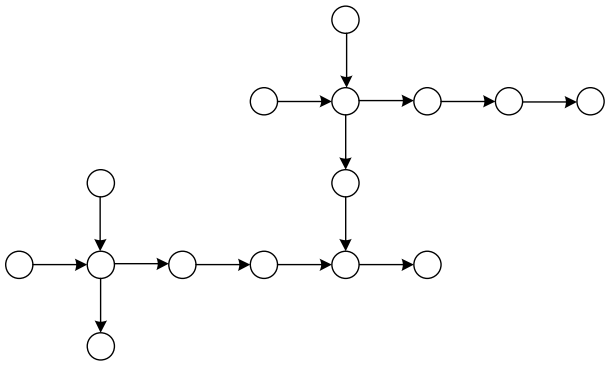

Fig. 3. An example network scenario. If channel switching is not allowed, all flows must be assigned the same channel.

To improve performance and route discovery success ratio, we need to allow some nodes to switch channels. Suppose in Figure 2, if node C can switch between two different channels, we can benefit from assigning different channels to these nodes, because transmissions such as A-B and C-H can take place simultaneously. However, allowing nodes to switch channels can lead to deafness problem, as we have explained in the previous subsection. To avoid the deafness problem, we allow nodes to switch channels with the following constraints.

- Two consecutive nodes in a path cannot switch channels. One of them has to stay in one channel.
- A node must notify its neighbors in a path whenever it switches channels.
- A node can only switch between a small number of channels (such as two), although many more channels are available.
- Nodes may not switch channels too frequently, such as per-packet basis.

The first two constraints are necessary to avoid the deafness problem, and the last two are for performance reasons. Since there is a channel switching delay of $80u$s, it is not desirable to have one node switch between too many channels, or switch channels frequently.

Returning to Figure 2, we can assign channel 1 for the route between A and E, and channel 2 for the route between F and I, and have node C switch between channel 1 and 2. Whenever C switches channels, C can broadcast messages on channel 1 and 2, so that the neighbors of node C knows which channel C is currently on.

Even with these constraints, allowing nodes to switch channels gives much more freedom to the routing protocol in selecting routes and assigning channels, and improves the route discovery success ratio. However, still there are cases where the route discovery may fail because of the constraints. In this case, forcing other routes to switch channels is inevitable. In the worst case, all flows may fall back to sharing a common channel, but still the performance does not go below that of a single channel protocol.

C. Channel Load Balancing

Until now, we have discussed different approaches of assigning channels and issues regarding these approaches. We choose the approach where channels are assigned to flows. Channel switching is allowed to improve the freedom of choice in route selection. Now the route discovery returns multiple routes to choose from, the protocol needs to select a route and also the operating channel for the route.

For single-channel routing protocols, routes are selected based on many different metrics such as hop distance, signal strength, degree of stability, and expected transmission time. For a multi-channel routing protocol, balancing load between available channels is another important goal.

Consider the scenario in Figure 4, where any pair of nodes can be reached in a single hop. Three flows can be established on node-disjoint routes. If there is no knowledge of traffic load on each channel, the protocol can randomly assign a channel for each flow. However, this may result in all three flows being assigned with the same channel. To be able to assign different channels to flows that are close to each other, nodes need to collect information on channel load. One possible solution is to

use HELLO messages, where each node periodically transmits HELLO messages on all channels, in a round robin manner. Sending HELLO messages too frequently may be too expensive. Also, since a node has to switch channels when sending HELLO messages, the node becomes temporarily deaf. On the other hand, if the period of sending HELLO messages is too long, the collected information may be stale. Thus, there is a trade off between overhead and the accuracy in collected channel load information.

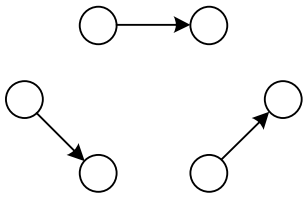

Fig. 4. An example network scenario with three node-disjoint flows. Random channel assignment may result in the same channel assigned for all flows.

The discussion presented in this section leads to the proposed protocol, called Multi-Channel Routing Protocol (MCRP), which is explained in more detail in the next section.

III. Multi-Channel Routing Protocol (MCRP)

In this section, we present the proposed Multi-Channel Routing Protocol (MCRP). MCRP is an on-demand routing protocol, and has many similarities with AODV [5]. However, MCRP benefits from multiple channels without requiring change in the underlying MAC protocol. It improves network performance by trying to allocate different channels to different flows, thus allowing simultaneous transmissions in a region. In the worst case, the performance of MCRP is at least comparable to AODV which is an on-demand single channel protocol. The protocol guarantees that a source will establish a route to the destination, if it could find a route in a single channel network with the same topology. MCRP assigns a common channel to all nodes in a flow, based on the discussion in the previous section. MCRP allows nodes to switch channels, but does not allow two consecutive nodes in a flow to switch channels, which causes deafness problem as also explained in the previous section.

Now we describe the protocol in detail. Since the MCRP shares many common features with the well-known AODV [5], we focus on the unique parts of MCRP, and omit the parts that are same as in AODV.

A. Node States

In MCRP, each node is in one of the four *node states* explained below. Suppose we have k channels. When a node is turned on, it may choose to stay on any of the k channels. When the node does not have any traffic it is responsible for transmitting, it can freely switch to other channels. We say that this node is in *free state*, and we call this node a *free node*. Once the node becomes a part of a traffic path and is assigned a channel, the node becomes a *locked node*. It is possible that two flows with different channels can intersect at a node. In this case, the node where two flows intersect needs to switch between two channels. This node is called a *switching node*. As explained later, MCRP prohibits a node to switch between more than two channels, due to excessive channel switching overhead. Also, as mentioned before, MCRP does not allow two consecutive switching nodes in a flow. Thus, the neighbors of a switching node in a flow become *hard-locked nodes*, and they must not be made switching nodes in order to prevent deafness problem. These four states are called *feasible states*, and a node must be in one of the four states. For example, consider the scenario with two flows in Figure 2. Flow 1 travels along the path A-B-C-D-E, and flow 2 uses the path F-G-C-H-I. Suppose the protocol assigns channel 1 to flow 1 and channel 2 to flow 2. Then node C becomes a switching node. Nodes A, C, F and I are locked nodes, because they are involved in flows. Nodes G, B, D, H are hard-locked nodes, because they are neighbors of C in a flow and must not become switching nodes. Finally, nodes J, K, L and M are free nodes. These node states affect route selection and channel assignment. The four node states are summarized in the following.

- free: The node does not have any flows and may freely switch to other channels.
- locked: The node has a flow on a certain channel.
- switching: The node is involved in multiple flows on different channels.
- hard-locked: The node has a flow on a certain channel, and it cannot be made a switching node.

B. Route Discovery

When a node has packets to send, it initiates a route discovery by broadcasting a Route Request (RREQ) packet. In a multi-channel network, nodes may stay in different channels. Thus, the RREQ packet must be broadcasted on all channels, as opposed to one channel in a single channel protocol. Suppose node S wants to find node D in a network with 3 channels. Also, suppose that node S was initially on channel 1. Then, S broadcasts RREQ in all channels, one by one in a round robin manner. More specifically, S broadcasts RREQ on channel 1, switch to channel 2, broadcasts RREQ on channel 2, and so on. After going through all channels, the node returns to the channel where it was originally staying on. Now when an intermediate node receives RREQ, it forwards the packet also on all channels. The intermediate nodes also return to their original channel after forwarding the packet. Since RREQ packets are forwarded on all channels, node D can receive the packet regardless of which channel it is on. Also, as the RREQ is forwarded, the nodes set up a *reverse path* to node S, as in AODV [5].

The number of RREQ packets transmitted is at most kn, where k is the number of channels, and n is the number of nodes in the network. For a single channel routing protocol which uses flooding for route discovery, the overhead is at most n. Thus, although the route discovery overhead of MCRP seems to be large, the overhead per channel is the same as that of a single channel protocol. Note that during the route request process, deafness problem may occur temporarily because nodes are switching channels.

The route entries of MCRP look like the Figure 5.

dest	*seqno*	*hops*	*nexthop*
channel	*active*	*expire*	*flags*

Fig. 5. A route entry in route table of MCRP.

In the route entry, all fields except *channel* and *active* fields are also in AODV. The *channel* field indicates the channel that the next hop node is on. So the node has to transmit a packet on the channel indicated in the *channel* field to reach the next hop. The *active* field is only relevant when the next hop node is a switching node (otherwise this field will have value 1). It indicates whether the next hop node is on the channel specified in the *channel* field. If this field has value 0, all packets that use this route must be buffered until the field becomes 1. How to manage the "active" field is explained later.

When a node sets up a reverse path to the source node, it needs to know which channel the next hop node is listening on. In order to provide this information, a node includes its operating channel (the channel which the node was on before broadcasting the packet) in the RREQ packet. This channel information is used by the nodes that received the RREQ packet to establish reverse path to the source node.

Finally, RREQ reaches the destination node D. Then, node D selects the channel which is to be used for the flow. The channel selection mechanism is explained later. After selecting a channel, node D sends a route reply (RREP) to node S using the reverse path.

As the RREP travels from node D to S, the nodes in the path switch channels and also node states according to the following. Suppose the selected channel is channel 1.

- free node: The node becomes a locked node and switches channel to channel 1.
- locked node: If the node is locked on channel 1, nothing changes. If the node is locked on another channel, it becomes a switching node between channel 1 and the original channel that this node was on.
- switching node: If channel 1 is one of the channels it is switching between, then nothing changes. Otherwise, the node drops the RREP packet. This is because MCRP does not allow a switching node to switch between three or more channels, as mentioned before.
- hard-locked node: If the node is locked on channel 1, nothing changes. Otherwise, the node drops the RREP packet, because hard-locked nodes are not allowed to become switching nodes.

Note that the RREP packet is dropped if the selected channel makes the node enter an infeasible state. However, if there are large number of flows in the network, a flow might only finds paths that requires one of intermediate nodes to go into an infeasible state. If all these routes are dropped, the source node may not be able to esta blish a route to the destination. To avoid this, we use the "force" mechanism which forces other routes to change, as will be explained later.

An example of route discovery process is illustrated in Figure 6. In the figure, there is a flow from X to Y on channel 2, and also a flow from M to N on channel 3. Suppose node S wants to find a route to node D. S starts a route discovery by broadcasting RREQ packets on all channels. After broadcasting on all channels, it returns to channel 1, where it was originally. When the RREQ is sent on channel 2, node A receives it and forwards it in all channels. Also, since node S includes its original channel in the RREQ, A knows that S is on channel 1 and sets up a reverse path to S. Node B does the same thing and finally when D receives the packet, D sends back RREP on the reverse path to S. Since each node knows which channel the receiver is on, it sends the packet on the receiver's channel. In this case, D sends on channel 3, B on channel 2, and A on channel 1.

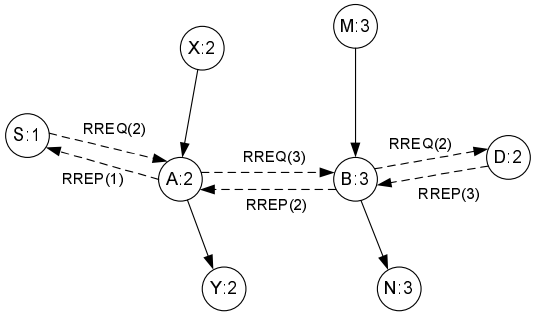

Fig. 6. A network scenario to illustrate route discovery process. The label "A:2" means that node A is on channel 2.

When the RREP packet reaches node S, the route is established and the data packets are transmitted on the selected channel. This is the basic skeleton for route discovery, but many of the details are left out for further explanation. They are described in the following subsections.

C. Channel Selection

When a destination node receives a RREQ packet, it needs to select a channel for the route. The channel selection algorithm must achieve two goals. First, no node in the path should go into an infeasible state. The channels that achieves this are called *feasible channels*. Second, among the feasible channels, the channel with the lowest load should be selected for the purpose of channel load balancing. For the first goal, the *channel table* is used, and for the second goal, the *flow table* is used. Both channel table and the flow table are carried in the RREQ packet to the destination.

1) Channel Table: The channel table records the state of nodes in the path from source to destination. As the RREQ packet is forwarded, each node updates the channel table included in the RREQ packet. Once the RREQ reaches the destination, the destination node selects the channel using the information provided from the channel table. The channel table contains a field for each channel as described in Figure 7.

ch_1	ch_2	…	ch_k

Fig. 7. The channel table included in the RREQ packet.

Initially, all fields are set to 0. The rules for updating the channel table depends on the node state. The rules are as follows.

- free node: No changes made to the channel table.

- locked node: If the locked channel is channel i, increment ch_i by one.
- switching node: if the two channels are channel i and j, increment ch_i and ch_j by one.
- hard-locked node: If the locked channel is channel i, increment ch_i by two.

To illustrate the use of channel table, we use Figure 6 again. In the figure, there are two flows: one from X to Y, and the other from M to N. Because of the flows, nodes X, A and Y are locked on channel 2, and nodes M, B and N are locked on channel 3. Nodes S and D are free nodes. When S sends an RREQ, it sends the channel table with zeros in all fields. If there are three channels, the channel table will be (0,0,0). When A forwards the RREQ, it adds on to channel 2 and the channel table becomes (0, 1, 0). Similarly, B updates the channel table to be (0, 1, 1). When D receives the RREQ, since it is a free node, the final channel table is (0, 1, 1). Using this channel table, node D chooses one channel according to the scheme explained later.

2) Flow Table: When there are multiple candidates for a selected channel, the channel should be chosen which makes the load balanced among channels. Since the protocol assigns the same channel for all nodes in a flow, the channel condition near the intermediate nodes must be considered as well as the channel condition near the destination, when the destination selects a channel. More specifically, the channel should be chosen which maximizes the throughput at the *bottleneck node*. Suppose there is a metric $x_c(i)$ denotes the interference level of channel c around node i in the path from S to D. Then the *path interference level* I_SD is:

$$I_{SD} = \min(\max(x_c(i), i \in P_{SD}), c \in C)$$

where P_{SD} is the path from node S to D and C is the set of candidate channels. The selected channel is the one which has the minimum I_{SD}.

In our protocol, we use the *number of flows in the neighborhood*. For each node, we count for each channel the flows that is going through itself and its neighbors. To collect the information from the neighbors we use HELLO mechanism. Each node transmits HELLO messages periodically, on all channels. Since a node can only transmit on one channel, it sends the packet in a round robin manner, as the RREQ packet. Again, this switching of channels may cause deafness problems, but the period of sending HELLO messages is long so that the problem does not affect the performance significantly.

In the HELLO packet, the node includes what channels it is on, and the number of flows in the channel. Using the HELLO messages and the flow state of itself, each node builds up its own *flow table*, which records the number of flows on each channel for itself and its neighborhood. We denote $F_c(i)$ as the number of flows in channel c around node i.

The flow table, which is included in the RREQ packet looks like Figure 8.

F_1	F_2	...	F_k

Fig. 8. The flow table included in the RREQ packet.

In the flow table, F_c means the number of flows for channel c. As the RREQ packet is passed, each node updates the flow table according to the following rule.

- For each c, if $F_c(i) > F_c$, then update F_c to be $F_c(i)$.

When the RREQ arrives at the destination, the destination node knows the interference level of each channel in the path.

We use Figure 6 once more to explain the flow table. Using the HELLO messages, each node builds up the flow table. Suppose we have three channels. Then after the two routes are established, A's flow table will look like (0,3,1), because two active neighbors and itself are on channel 2, and one active neighbor is on channel 3. Node S's flow table will be (0,1,0), and node B's table will be (0,1,3).

In the following, the flow table indicates the flow table in the RREQ packet, not in a node. When S sends an RREQ, it updates the flow table to (0,1,0). When A receives it, it updates the flow table in RREQ to (0,3,1). Now when the RREQ comes to B, B updates the flow table to (0,3,3), following the rule above. Finally, D updates it to (0,3,3) which is the final flow table used in the channel selection.

3) Channel Selection Algorithm: When the destination node receives a RREQ packet, it needs to select a channel for the flow and sends back the RREP packet. The channel selection is performed using the channel table and the flow table included in the RREQ packet.

First, the destination node tests to see if the route is feasible, meaning that it does not create any two consecutive switching nodes in a flow or assign more than two channels for a node. The test is done using the channel table. The route is determined to be infeasible if:

- Multiple channels have values greater or equal to 2.
- More than two channels have values greater or equal to 1.

If any of these two conditions are met, then the route is considered infeasible. Then the route is either dropped, or is still used if the protocol uses "force mechanism" explained later.

If the route is considered feasible, then the destination node chooses the channel considering the flow table, according to the following rules.

- If a channel has a value greater or equal to 2, this channel has to be selected.
- If two channels have a value 1, then one of these two channels which has the minimum interference level is selected.
- If only one channel has a value 1 and all other channels have 0's, then among all channels, the one with the minimum interference level is selected.

Returning to the example in Figure 6, node D has the final channel table of (0,1,1), and the final flow table of (0,3,3). According to the above rules, this route is feasible, because only two channels have one as their values. Also, either channel 2 or channel 3 has to be selected because they both have a value 1. Now since two channels have the same interference level of 3, then one is chosen at random.

4) Delayed Reply: In a single channel on-demand routing protocol such as AODV, the destination node replies to the first route request and disregards all others. This is because the first route request to reach the destination has presumably taken the route with the least delay. Routing protocols that value other metrics more than delay, or protocols that maintain multiple routes used *delayed reply* mechanism, where the destination

does not reply to the first request but wait for some time hoping that more RREQs arrive through other paths. Then the destination can choose one or more routes according to the metrics they use.

Also, in AODV, the intermediate nodes only forward the route request once, due to the same reason that the first route request is taking the path of least delay. However, other protocols allow the intermediate nodes to forward the route requests after the first one, if it has a better metric than the first one.

MCRP also uses the delayed reply mechanism. When the destination receives the first RREQ, it sets up a timer and waits for other route requests to arrive. MCRP also allows the intermediate nodes to forward the request after the first one, if *the route is feasible and the path interference level is lower* than the previous requests. If the destination receives multiple RREQs, it chooses the RREQ which results in a selected channel with minimum interference level, and ignores other RREQs.

D. Packet Forwarding and Channel Switching

Once a route has been established through the route discovery process, the source node can begin sending data packets to the next hop node in the path. Since MCRP assigns a common channel to all nodes in a flow, nodes can send packets on the selected channel without switching to another channel. However, communicating with a switching node is more complicated, since it switches channels from time to time to support multiple flows in different channels. A node may fail to send packets to a switching node, if the switching node is listening on another channel. Furthermore, a node may falsely think the route is broken after several tries, if the switching node stays on another channel for a long time, causing significant performance loss. So the neighbors of a switching node must know whether the switching node is available on the channel they are listening on.

To achieve this, a switching node uses LEAVE/JOIN messages to inform its neighbors of its channel status. Suppose a switching node B switches channels from 1 to 2. Before switching channels, B sends a LEAVE message on channel 1. The neighbors listening on channel 1 receive the LEAVE message and reset the 'active' flag in the route entries having node S as the next hop. Then B switches its channel to channel 2. After switching channels, B sends a JOIN message on channel 2. The neighbors on channel 2 receive this message and set the 'active' flag in the route entries having node S as the next hop.

When a node has packets to send to a switching node which is currently listening on another channel, the routing protocol needs to buffer the packets until it receives a JOIN message from the switching node. For this reason, the routing protocol maintains a separate buffer for keeping the packets waiting for the switching node. When the node receives a JOIN message from the switching node, the packets in this buffer are sent with a higher priority then other packets.

A switching node needs to decide the duration of time it stays on channel before switching to another channel. A node may potentially switch channels after sending each packet, but this results in performance loss due to the overhead of channel switching delay and LEAVE/JOIN message overhead. On the other hand, if a node stays on one channel too long, the delays for the packets on the other channel will be too high. So the duration for staying in one channel must be determined intelligently. This duration can be determined according to an estimation of traffic load in each channel. The adaptive algorithm for determining the duration of time a switching node stays in one channel is outside scope of this paper and is one of our future work. In the simulations in this paper, we use fixed duration of 50ms regardless of traffic load.

E. Force Mechanism

With the basic scheme of MCRP described above, it is possible that the destination node receives one or more RREQs, but all potential routes are determined to be infeasible. If the destination drops all these routes, it will result in the source failing to find a route although there is a path between the source and the destination.

In this case, the destination may still choose to select one route and send reply back, with the "force" flag set in the RREP packet. This option is called *force mechanism* and it is an optional feature of our protocol. The force mechanism guarantees that a route can be found if there is a path between source and destination.

If an intermediate node receives a RREP with the "force" flag on, then it is forced to choose the channel specified in the RREP, even if it is locked on another channel. The node selects channels according to the following rules. Suppose the RREP specifies channel x.

- Free node: The node becomes a locked node, and it is locked on channel x.
- Locked or Hard-locked node: If it is already locked on channel x, do nothing. If it is locked on another channel, send RERR for the flows that were on the other channel, and switch to channel x. The node state remains unchanged.
- Switching node: If one of its operating channels is channel x, then do nothing. If not, then choose one of its operating channel and send RERR for the flows on that channel, and replace that channel with channel x in the set of its operating channels. The node state remains unchanged.

The rules do not allow a locked node to become a switching node for a "forced" route, because it may cause two consecutive switching nodes in a flow.

There is at least one flow that loses the route because of this forced route. The route is considered to be broken, and the RERR is forwarded to the sources as noted above. Then to recover from route error, source has to perform route discovery again. It is possible that two flows might force each other to break in the process of finding a route. To avoid the oscillation, nodes that have a route created by a force mechanism do not accept another RREP with the force flag on for a certain duration of time after it is created.

With the force mechanism, we can guarantee that a route can be eventually found if there is a path between source and destination. In the worst case, all routes will be converged into one common channel.

F. Route Maintenance

The route maintenance of MCRP is mostly similar to AODV [5]. When a node obtains a route, it sets up a timer for the route so that if the route is not used for a certain period of time, it is considered to be stale and is deleted from the route table. Whenever a route is used, it is "refreshed" meaning that the timer is reset. Due to node failures and mobility, a route may break while in use. If the MAC layer tries to transmit a packet several times without success, it tells the routing protocol that the link is broken. When a node determines that a link is broken, it removes the route from the route table and notifies other nodes that use the broken link to reach the destination (similar to AODV). The notification is done by sending a RERR packet. To reduce the cost of sending RERRs, a node keeps the *precursor list*, which includes nodes that are placed before the current node in the path from the source to destination. So only when a node has a precursor for the broken route, it transmits RERR to notify them of the route failure.

In MCRP, node states must be changed according to the change in the routes. When a route is removed from a node, this may affect the state of the nodes. For example, if all routes are removed from the route table of a node, the node becomes a free node. A switching node may become a locked node if all routes on one channel are removed. The rules for changing the node states is as follows.

- locked node: If all routes are removed, the node becomes a free node.
- hard-locked node: If all routes are removed, the node becomes a free node. If all routes that have a switching node as its next hop are removed, then the node becomes a locked node.
- switching node: If all routes in one channel are removed, then the node becomes a locked node.

IV. Performance Evaluation

We have evaluated the performance of our protocol MCRP through simulations using ns-2 simulator [6]. The metric we measure is the aggregate throughput over all flows. For each set of simulations, we run MCRP with 2, 3, and 4 channels, and also AODV with a single channel for the purpose of comparison.

We vary several parameters to see how various factors affect the performance of MCRP. The varying parameters used are: number of flows, flow rate, and connection pattern. The connection pattern is the set of source-destination pairs, and the connection pattern may affect the performance of MCRP significantly because channel allocation is done per flow.

For the traffic, we use constant bit rate (CBR) traffic, over UDP. The packet size is 512 bytes. All channels bit rate is 11Mbps. The network area is a square of 1000m x 1000m, and the transmission range of each node is approximately 250m. IEEE 802.11 DCF is used as the MAC protocol without any change.

In the first simulation, we measured the aggregate throughput varying the number of flows. 100 nodes are randomly placed in the area. Each flow generates traffic at 4Mbps. Since the flow rate is 4Mbps, the channel bandwidth is quickly saturated, as the number of flows is increased. The result is shown in Figure 9. The figure shows that MCRP can sometimes improve the throughput by a factor of 4 with 4 channels. Because of the routing protocol overhead and flow-level channel allocation, the performance of MCRP cannot reach the factor k improvement over a single channel protocol, where k is the number of channels. However, since the multiple channels distribute contention over the channels, the collision rate is reduced. That is why sometimes the throughput of MCRP with k channels is more than k times the throughput of a single channel protocol.

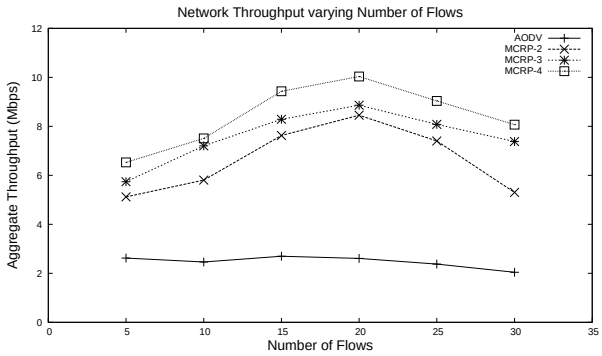

Fig. 9. Network Throughput varying Number of Flows

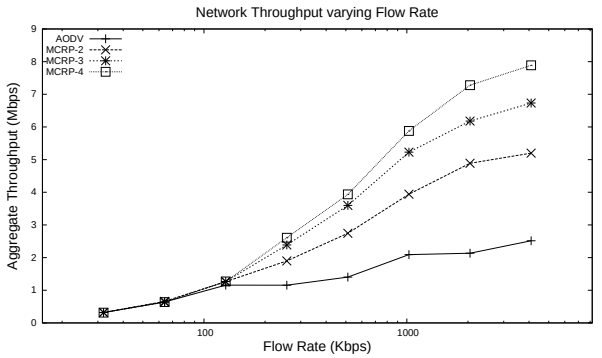

Fig. 10. Network Throughput varying Flow Rate

In the next simulation, we varied the rate of each flow from 32Kbps to 4096Kbps. There are 50 nodes randomly placed in the area, and 10 flows having the same traffic generation rate. The results are shown in Figure 10. When the traffic load is low, having multiple channels does not make a difference because the total traffic is less than the bandwidth of a single channel. With low load, AODV can do a slightly better because of the overhead in MCRP, but as the graph shows, there is no significant difference. As the flow rate increases, the throughput of AODV increases slowly towards the limit of a single channel. As the traffic load becomes large, the performance improvement of MCRP over AODV becomes more significant.

Finally, we simulated the protocol in different scenarios to see how the network topology and the location of source and destination affect protocol performance. In each scenario, 50 nodes are randomly placed in 1000m x 1000m area, and there are 10 flows with the traffic generation rate of 4Mbps. As shown in Figure 11, in most of the scenarios, MCRP with k channels achieves the throughput of a little less than k times the through-

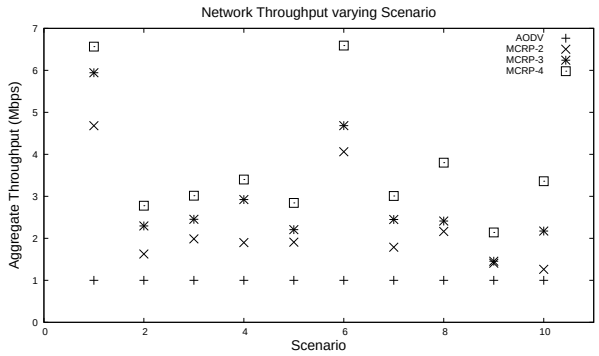

Fig. 11. Network Throughput varying Scenario

put of AODV, a single channel protocol. But sometimes the performance of MCRP is significantly higher than AODV, even more than factor of k, because MCRP reduces the chance of collision by distributing the contention of flows among channels. In 2 out of 10 scenarios, MCRP with 2 channels achieves throughput of more than twice the throughput of AODV.

The simulation results show that MCRP significantly improves the network throughput by utilizing multiple channels, even with a single transceiver at each node.

V. Related Work

In this section, we review the relevant work in this area. We first review multi-channel MAC protocols, and then review routing and channel assignment protocols.

Nasipuri et al. [7] proposed a multi-channel protocol with the assumption that a node can simultaneously listen to all channels. When a node has a packet to send, it senses carrier on all channels and selects an idle channel. This protocol requires nodes to have N transceivers, where N is the number of available channels. This protocol is extended to select channels according to signal strength in [8].

Wu et al. [4] proposed a MAC protocol that assigns channels dynamically, in an on-demand style. This protocol requires each node to have two transceivers. Between two transceiver, one is always listening to a predefined common channel, and the other one can switch between data channels. The control messages are exchanged in the control channel so that every node can receive the control messages. Data channels are assigned using a negotiation between the sender and the receiver. The protocol was extended in [9] to include power control with the channel assignment. Jain et al. [10] proposed a similar approach, assuming two transceivers in each node. In this protocol, the receiver selects a channel according to the channel condition at the receiver side.

The authors have proposed a multi-channel MAC protocol that works with a single transceiver [11]. The idea is to have an interval where every node listens to a common channel and negotiate channels. After the interval, nodes switch to the negotiated channels and start communicating on the channel. Since this protocol requires that periodically every node must listen to a common channel, temporal synchronization is required for this protocol to work.

Bahl et al. [12] proposes a protocol, in which nodes switch from channel to channel according to a sequence. Two nodes can exchange messages when their channel schedules overlap. If a node wants to initiate a connection with another node, it switches its channel hopping sequence to match that of the receiver, so that two nodes stay on the same channel for a long time.

Now we review the routing and channel assignment protocols. Shacham and King [13] proposed routing protocols for multi-channel networks. The paper proposes different schemes for different scenarios. In the first scenario, every node has a single transceiver. In this case, every node broadcasts its routing information in all channels. After gathering the route information, a sender switches to the destination's channel and start transmitting packets. This scheme assumes that every node has at least one neighbor for every channel. This may be true for dense networks, but is often not true in a sparse networks. In the second scenario, there are nodes with multiple radios. Then they can act as relays and forward packets from one channel to another. This scheme requires enough number of nodes with multiple radios to establish routes between any pair of nodes in the network.

Raniwala et al. [2] proposed a centralized channel assignment algorithm assuming that each node has two transceiver. With N channels available, this protocol assigns these channels to nodes so that the network is not partitioned and bandwidth can be efficiently allocated. Adya et al. [14] proposed a link-layer protocol that manages underlying multiple interfaces. The protocol performs channel selection, so that the overall channel utilization is optimized.

Chandra et al. [15] proposes a software layer which enable a single wireless card to connect to multiple networks. Also, the paper proposes algorithm for switching between networks and protocol for buffer management. Finally, Draves et al. [1] proposes a metric for multi-channel networks combining expected transmission time and channel diverseness.

VI. Conclusion and Future Work

In this paper, we considered a network layer approach for utilizing multiple channels in wireless ad hoc networks, assuming a single-channel MAC protocol and single transceiver at each node. We identified that assigning channels to nodes can result in significant performance degradation due to the deafness problem. On the other hand, assigning nodes to flows without allowing dynamic channel switching can result in low utilization of channels and also reduce the route discovery success ratio. Finally, to avoid deafness problem, two consecutive nodes in a path should not be both switching nodes.

Based on the discussions, we proposed a MCRP, a routing protocol that utilizes multiple channels to improve network throughput. MCRP works with a single transceiver, and a single-channel MAC protocol such as IEEE 802.11 DCF. Since MCRP does not require additional hardware or a new MAC protocol, it can be easily deployed to currently used devices. The simulation results show that MCRP improves network throughput substantially by efficiently allocating channels to flows.

There are several issues that are not addressed in this paper. First, as described in Section III, switching nodes maintain a separate queue for each channel. We assumed in this paper

that nodes switch between channels according to a fixed period. However, the interval that the node stays in each channel can be determined according to number of pending packets in each queue. Also, the interval can be determined according to the observed channel condition. For example, a node can increase the time spent in a channel if the channel condition is good, and quickly switch to a different channel if the channel condition is bad. This issue requires a further research. Second, MCRP assigns a common channel to all nodes in a flow. This may not be efficient for flows with a large number of hops, because channel conditions may vary in different parts of the flow. Assigning different channels to nodes in a flow while avoiding deafness problem is another issue we are planning to address in the future. Finally, we are planning to investigate a cross layer approach, where MAC and network layer cooperate with each other to achieve higher performance.

ACKNOWLEDGMENTS

This research was supported in part by Motorola Center for Communication. The authors would like to thank Jeff Bonta for his comments on this work.

REFERENCES

[1] Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *ACM Mobicom*, 2004.

[2] Ashish Raniwala, Kartik Gopalan, and Tzi-cker Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, April 2004.

[3] R. Choudhury and Nitin Vaidya, "Deafness: A mac problem in ad hoc networks when using directional antennas," in *IEEE ICNP*, 10 2004.

[4] Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu, "A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks," in *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, 2000.

[5] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," in *Ietf RFC 3561*, July 2003.

[6] VINT Group, "UCB/LBNL/VINT network simulator ns (version 2)," .

[7] A. Nasipuri, J. Zhuang, and S.R. Das, "A multichannel csma mac protocol for multihop wireless networks," in *WCNC*, September 1999.

[8] A. Nasipuri and S.R. Das, "Multichannel csma with signal power-based channel selection for multihop wireless networks," in *VTC*, September 2000.

[9] S.-L. Wu, Y.-C. Tseng, C.-Y. Lin and J.-P. Sheu, "A Multi-Channel MAC Protocol with Power Control for Multi-Hop Mobile Ad Hoc Networks," *The Computer Journal*, vol. 45, pp. 101–110, 2002.

[10] N. Jain, S. Das, and A. Nasipuri, "A multichannel csma mac protocol with receiver-based channel selection for multihop wireless networks," in *IEEE International Conference on Computer Communications and Networks (IC3N)*, October 2001.

[11] Jungmin So and Nitin H. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *Mobihoc*, 2004.

[12] Paramvir Bahl, Ranveer Chandra, and John Dunagan, "Ssch: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks," in *ACM Mobicom*, 2004.

[13] N. Shacham and P. King., "Architectures and performance of multichannel multihop packet radio networks," *IEEE Journal on Selected Area in Communications*, vol. 5, no. 6, pp. 1013– 1025, July 1987.

[14] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou, "A multi-radio unification protocol for ieee 802.11 wireless networks," in *IEEE International Conference on Broadband Networks (Broadnets)*, 2004.

[15] Ranveer Chandra, Paramvir Bahl, and Pradeep Bahl, "Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card," in *IEEE Infocom*, Hong Kong, March 2004.