Routing and Link-layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks*

Pradeep Kyasanur[a] *kyasanur@uiuc.edu*
Nitin H. Vaidya[b] *nhv@uiuc.edu*
[a]Dept. of Computer Science, and Coordinated Science Lab, University of Illinois at Urbana-Champaign
[b]Dept. of Electrical and Computer Eng., and Coordinated Science Lab, University of Illinois at Urbana-Champaign

Wireless technologies, such as IEEE 802.11a, that are used in ad hoc networks provide for multiple non-overlapping channels. Most ad hoc network protocols that are currently available are designed to use a single channel. However, the available network capacity can be increased by using multiple channels. This paper presents new protocols specifically designed to exploit multiple channels. Our protocols simplify the use of multiple channels by using multiple interfaces, although the number of interfaces per host is typically smaller than the number of channels. We propose a link layer protocol to manage multiple channels, and it can be implemented over existing IEEE 802.11 hardware. We also propose a new routing metric for multi-channel multi-interface networks, and the metric is incorporated into an on-demand routing protocol that operates over the link layer protocol. Simulation results demonstrate the effectiveness of the proposed approach in significantly increasing network capacity, by utilizing all the available channels, even when the number of interfaces per host is smaller than the number of channels.

I. Introduction

Wireless technologies, such as IEEE 802.11a [1], provide for multiple non-overlapping channels. Multiple channels have been utilized in infrastructure-based networks by assigning different channels to adjacent access points, thereby minimizing interference between access points. However, multi-hop wireless networks have typically used a single channel to avoid the need for co-ordination between adjacent pair of nodes, which is necessary in a multi-channel network. For meeting the ever-increasing throughput demands of applications, it is important to utilize all of the available spectrum, and this motivates the development of new protocols specifically designed for multi-channel operation.

Wireless hosts have typically been equipped with one wireless interface. However, a recent trend of reducing hardware costs [2] has made it feasible to equip nodes with multiple interfaces. Nevertheless, it is still expensive to equip a node with one dedicated interface for each channel, as the number of channels may be large. Even if each channel does not have a dedicated interface, currently available commodity wireless interfaces (such as IEEE 802.11 wireless interface cards) can be *switched* from one channel to another, albeit at the cost of a switching latency, thereby allowing all channels to be potentially utilized. Thus, it is of practical interest to develop protocols for the scenario wherein the *number of interfaces per node is smaller than the number of channels.*

When c channels are available, but each node has only $m < c$ interfaces, one possibility is to keep the m interfaces fixed on some m channels only [3], which implies the remaining $c - m$ channels are not used. Therefore, such an approach may waste a lot of channels (especially when $m \ll c$). Our goal is to utilize all the available channels even when there are few interfaces, by switching the available m interfaces among the c channels. While this is feasible in theory [4], achieving this in practice raises many challenges [5]. For example, for the full utilization of available channels, it is desirable to have different nodes communicating (in parallel) on different channels. However, two adjacent nodes can communicate with each other only when they have at least one interface on a common channel. Thus, there is an inherent trade-off between the need to use different channels for increasing capacity, and the need to use a common channel for ensuring connectivity between nearby hosts.

In this paper, we present a new link layer protocol for utilizing multiple channels. The link layer protocol is based on a novel interface assignment strategy [5] that classifies available interfaces into "fixed" and "switchable" interfaces. Fixed interfaces stay on specified "fixed channels" (can be different for differ-

*This research was supported in part by NSF under grant ANI-0125859, and a Vodafone Graduate Fellowship.

ent nodes) for long intervals of time, while switchable interfaces can be switched more frequently, as necessary, among the non-fixed channels. By distributing fixed interfaces of different nodes on different channels, all channels can be utilized, while the switchable interface can be used to maintain connectivity. A key advantage of the link-layer protocol is that it can be implemented using off-the-shelf hardware.

Any existing ad hoc routing protocol can be used over our proposed link-layer solution. However, the throughput of a route that uses a single channel on all hops can be substantially smaller than a route that uses multiple channels (i.e., a "channel diverse" route), on account of *self interference* along the route. Furthermore, for utilizing all the available channels, interface switching may be required, and the cost of interface switching has to be accounted for, when selecting routes. We propose a new multi-channel routing (MCR) metric that *selects channel diverse routes*, while accounting for interface switching cost. We evaluate our proposed MCR metric with an on-demand source routing protocol (similar to DSR [6]). Simulation results show that a multi-channel network that uses the combined link layer and routing solution has substantially higher throughput than a single channel network.

The rest of the paper is organized as follows. We describe related work in Section II, and formulate the multi-channel, multi-interface problem in Section III. Sections IV and V describe the details of the proposed approach. We evaluate our proposal in Section VI. We discuss possible extensions to our proposal in Section VII, and conclude in Section VIII.

II. Related Work

The theoretical benefits of using multiple channels and multiple interfaces, along with centralized algorithms for achieving the benefits, have been presented recently in [4, 7, 8]. In this paper, we present distributed protocols for utilizing multiple channels.

Several researchers have proposed MAC protocols for utilizing multiple channels (c.f., [9, 10, 11, 12]). These multi-channel protocols require changes to existing standards, such as IEEE 802.11, and therefore cannot be deployed by using commodity hardware. Adya et al. [13] propose a link-layer solution for striping data over multiple interfaces, but their solution is designed for the scenario where the number of interfaces per host is equal to the number of channels.

Existing routing protocols for multi-hop networks, such as DSR [6] and AODV [14], can support multiple interfaces at each node. However, those protocols typically select shortest-path routes, which may not be suitable for multi-channel networks [3]. Shacham et al. [15] have proposed an architecture for multi-channel wireless networks that uses a single interface. However, the proposed architecture may affect network connectivity and requires complex coordination. So et al. [16] have proposed a routing protocol for multi-channel networks that uses a single interface at each node. Their routing protocol requires tight synchronization between nodes, while our proposed solution works with loose synchronization between nodes.

Bahl et al. [17] have proposed SSCH, a link layer solution that uses a single interface. SSCH can be extended to utilize multiple interfaces as well. SSCH requires individual nodes to hop among channels based on a well published schedule. A node stays on a channel for a specified slot time (10 ms), and the delay incurred in switching an interface from one channel to another is expected to be small (80 μs). A key requirement for achieving good performance with SSCH is to use short slot times, which in turn requires fast interface switching. However, currently available commercial hardware require at least a millisecond to switch channels, which is an order of magnitude higher than what SSCH assumes. A key design goal of our solution is to realize good performance even when interface switching delay is fairly large, thereby allowing the solution to be implemented with currently available hardware. In addition, our solution provides a joint link layer and routing solution that is carefully designed to exploit multiple interfaces, and can provide good performance in multi-hop networks.

There are a few routing proposals specifically designed for multi-channel *and* multi-interface wireless networks. Raniwala et al. [18, 19] propose routing and interface assignment algorithms for static networks. Similar to our proposal, they also consider the scenario wherein the number of available interfaces is less than the number of available channels. However, their solution is designed specifically for use in those mesh networks where all traffic is directed toward specific gateway nodes. In contrast, our proposal is designed for more general ad hoc networks (as well as mesh networks with significant intra-mesh traffic), where potentially any node may communicate with any other node.

Draves et al. [3] have proposed a new routing metric, called WCETT, for multi-channel ad hoc networks that ensures "channel diverse" routes are selected. WCETT has been designed with the assumption that

the number of interfaces per node is *equal* to the number of channels used by the network. In contrast, our proposal is designed to handle the more general scenario where the number of available interfaces may be smaller than the number of available channels, and *interface switching* is required to utilize all the channels. Therefore, we propose a link layer protocol to enable the use of all channels, and we develop a new metric called MCR, which extends WCETT metric, to complement the link layer protocol.

III. Preliminaries

III.A. Number of orthogonal channels

IEEE 802.11a standard provisions for 12 non-overlapping channels in the US. When a single node is equipped with multiple interfaces, it has been noted [3] that communication on different interfaces using adjacent non-overlapping channels may interfere. Thus, the number of available orthogonal channels (i.e., channels that can be used simultaneously) may be smaller than the number of non-overlapping channels. Recently, Raniwala et al. [19] have experimentally shown that when distance separation between interfaces is increased, the interference between interfaces is reduced, allowing more channels to be used simultaneously. We have also performed measurements using Atheros-based [20] 802.11a cards, which suggest that 5 or 6 channels (e.g., channels 36, 48, 64, 149, 161 in IEEE 802.11a) are orthogonal when using existing interface hardware. Furthermore, future hardware that employs better filters to reduce adjacent channel interference may allow any pair of non-overlapping channels to be simultaneously used.

In our simulations, we evaluate the performance of our protocols both with 5 orthogonal channels (total orthogonal channels available with current hardware), and with 12 orthogonal channels. Our simulations indicate that even when only 5 channels are available, significant performance improvements are possible.

III.B. Interface switching latency

Switching an interface from one channel to another incurs some delay, *switchingDelay*, which may be non-negligible. In the current literature, estimates for *switchingDelay* (for switching between channels in the same frequency band) with commodity IEEE 802.11 hardware are in range of a few milliseconds [21, 19]. It is expected that with improving technology, the switching delay can be reduced to a few tens of microseconds [17]. Protocols that utilize interface switching need to be flexible enough to accommodate a range of switching delays. Most of our simulation results use a value of 1 ms for switching delay, but our protocol offers good performance even with larger values of switching delay.

III.C. Problem formulation

The protocols proposed in this paper are designed for an ad hoc multi-hop wireless network. Nodes in the network can be mobile. We assume that the typical traffic pattern involves communication between arbitrary pairs of nodes. We also assume that each host has at least two interfaces. We define the goals of a multi-channel, multi-interface solution as follows:

1. Improve network capacity by fully utilizing all the available channels, *even if the number of interfaces per host is smaller than the number of channels*. The solution must accommodate different number of channels and interfaces.

2. Ensure that a network that is connected when using a single common channel, continues to be connected when multiple channels are used.

3. Allow implementation on existing 802.11 hardware. This requires solutions to offer good performance even if switching delay is fairly large

III.D. Solution approach

We develop a link layer protocol (Section IV) to manage the use of multiple interfaces, and a routing protocol (Section V) that interacts with the link layer protocol to select good routes. Such a separation of functionality is used to simplify protocol design. Interface switching can occur on the timescales of a few packet transmissions; hence it is beneficial to incorporate interface management at the link layer, as part of the kernel. Route selection happens on larger timescales (often hundreds of packet transmissions or more), and it is beneficial to implement it separately, possibly as an user space daemon. A further benefit of this approach is that we may operate an existing routing protocol over the link layer protocol, since the link layer completely hides the complexity of managing multiple channels and interfaces from the higher layers.

IV. Link layer protocol

An interface assignment strategy is required to assign interfaces to specific channels when the number of available interfaces is smaller than the number of available channels. Furthermore, for utilizing

all the available channels, interfaces may have to be switched, and a protocol is necessary to decide when to switch an interface from one channel to another. The protocol has to ensure that the neighbors of a node X can communicate with it on-demand, which requires all neighbors of X to be always aware of at least one channel on which X has an interface. We have provided a classification of interface assignment strategies in [5]. In [5], we have also presented a new interface assignment strategy, which is briefly described in the next sub-section. We then describe the new link layer protocol proposed in this paper for implementing the strategy.

IV.A. Background: Interface assignment strategy [5]

Suppose that M interfaces are available at each node. The available interfaces are divided into two subsets.

1. *Fixed Interfaces:* Some K out of M interfaces at each node are assigned for long intervals of time to some K channels. We designate these interfaces as "fixed interfaces", and the corresponding channels as "fixed channels".

2. *Switchable Interfaces:* The remaining $M - K$ interfaces are dynamically assigned to any of the remaining $M - K$ channels (over short time scales), based on data traffic. These interfaces are designated as "switchable interfaces", and the channels to which a switchable interface may be assigned to are called "switchable channels".

Different nodes may assign their K fixed interfaces to a different set of K channels. It is possible for each node to use a different value of K and M, and it is also possible to vary K with time. A node X with a packet to send to a node Y has to send the packet on a fixed channel of Y. In the rest of this section, we describe a new link layer protocol for implementing this strategy, and explain how the protocol can utilize all the available channels.

IV.B. Protocol for communication between nodes

For simplifying the description of the protocol, we assume that $M = 2, K = 1$ for all nodes, i.e., there is one fixed interface, and one switchable interface (although the protocol proposed next can be extended for any values of $M \geq 2$ and $1 \leq K < M$). Each node at initialization designates one interface as its fixed interface, and the second interface as the switchable interface. Fixed interface of a node is assigned to a "fixed channel" for long intervals of time. Each node maintains a *NeighborTable* containing the fixed channels being used by its neighbors (the details on constructing this table are in Section IV.C).

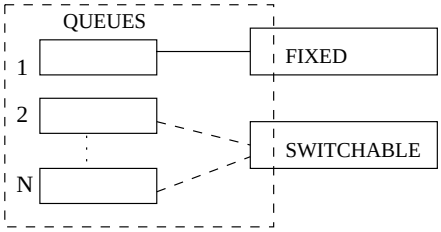

Figure 1: Illustration of queues associated with N channels and 2 interfaces

Inserting packets into channel queues:

Each channel is associated with a packet queue, as shown in Figure 1. If an unicast packet is received at the link layer for transmission, the fixed channel of the destination of the packet is looked up in the *NeighborTable*, and the packet is added to the corresponding channel queue.

In single channel networks, a packet broadcast on the channel can potentially be received by all neighbors of the transmitter. However, when multiple channels are being used, a packet broadcast on a channel is received only by those nodes listening to that channel. Many higher-layer protocols (e.g., routing protocols) require broadcast packets to be received by all nodes in the neighborhood. Such neighborhood broadcast is supported in our protocol by transmitting the broadcast packet separately on every channel. A copy of the broadcast packet is added to each channel's queue, and sent out when that channel is scheduled for transmission by the protocol.

Although sending a copy on each channel increases the cost of broadcast with increasing number of channels, the *cost per channel* remains same. For example, in a M channel network, each broadcast involves sending M copies of the packet on M channels, resulting in only 1 packet per channel. Thus, the overhead per channel is the same as in a single channel network. However, too many broadcasts may still be detrimental to performance, since the switchable interface has to frequently switch channels to transmit each copy of a broadcast packet, degrading performance. It is part of our future work to use *partial broadcasts* [17], wherein broadcast packets are sent out only on a subset of available channels. The partial broadcast approach can reduce overheads, though at the cost of reducing connectivity.

Servicing channel queues:

The fixed interface transmits packets queued up for transmission on the fixed channel. Packets are transmitted on all other channels using the switchable interface. When the switchable interface is switched to a new channel, it is always switched to the channel with the oldest queued data. This policy ensures fairness. The switchable interface changes channels only when there are packets queued for another channel, and one of the following two conditions hold:

1. The switchable interface is on a channel with an empty queue.

2. The switchable interface has been on a channel for more than *MaxSwitchTime* duration. This condition prevents starvation of other queues.

Using a small value of *MaxSwitchTime* increases switching overhead (although the overhead increases only if data is present for multiple channels), while using too large a value increases end-to-end latency. In our simulations, we set *MaxSwitchTime* to be at least five times the switching delay (which is sufficient to allow a few packet transmissions), and the simulated performance is good. Experimenting with other values is part of our future work.

When an interface is switched to a new channel, the virtual NAV on the new channel (set from overheard RTS-CTS transmissions) may not be correct because the node may have missed earlier RTS-CTS transmissions. If RTS-CTS is enabled in the network, then the interface defers for one maximum sized packet transmission. This strategy ensures that ongoing transmissions are protected from interference. If RTS-CTS is not enabled, then the interface has to only defer until the channel is idle. In both cases, after deferring, the interface may begin transmitting data (after following MAC backoff rules, etc.).

Example of protocol operation:

Figure 2 illustrates the protocol operation. Assume that node A has packets to send to node C via node B. Nodes A, B, and C have their fixed interfaces on channels 1, 2, and 3, and switchable interfaces on channels 3, 1, and 2 respectively. In the first step, node A *switches* its switchable interface from channel 3 to channel 2, before transmitting the packet, because channel 2 is the fixed channel of node B. Node B can receive the packet since its fixed interface is always listening to channel 2. In the next step, node B switches its switchable interface to channel 3 and forwards the packet, which is received by node C using its fixed interface. Once the switchable interfaces

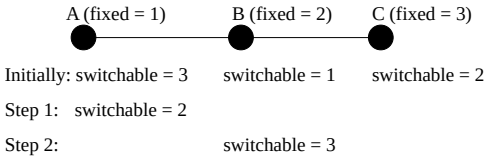

Figure 2: Example of link layer protocol operation with 3 channels, 2 interfaces

are correctly set up during a flow initiation, there is no need to switch the interfaces for subsequent packets of the flow (unless the switchable interface has to switch channels for sending packets of a different flow).

IV.C. Managing fixed interface

Fixed interface management involves two components - choosing the channel to be assigned to the fixed interface, and informing neighbors about the channel being used by the fixed interface. The link layer protocol has to ensure that fixed interfaces of nodes in a neighborhood are distributed across different channels. For example, suppose a node A uses channel 1 for the fixed interface. Then, all transmissions directed to A will be on channel 1. For balancing the usage of channels, it is beneficial if other nodes in the neighborhood use a different channel for their fixed interface.

We propose a localized protocol for fixed interface management. Recall that each node maintains a *NeighborTable* containing the fixed channels being used by its neighbors. Nodes also maintain a *ChannelUsageList* containing a count of the number of nodes in its two-hop neighborhood using each channel as their fixed channel. Initially, a node chooses a random channel for its fixed interface.

Hello packet exchange:

Periodically, each node broadcasts a "Hello" packet on every channel. The hello packet contains the fixed channel being used by the node, and its current *NeighborTable*. When a node receives a hello packet from a neighbor, it updates its *NeighborTable* with the fixed channel of that neighbor. The *ChannelUsageList* is updated using the *NeighborTable* of its neighbor. Updating *ChannelUsageList* with each neighbor's *NeighborTable* ensures that *ChannelUsageList* will contain two-hop channel usage information. An entry which has not been updated for a specified maximum lifetime is removed. This ensures that stale entries of nodes that have moved away are removed from the *NeighborTable* and *ChannelUsageList*.

The frequency of hello packet exchange depends on

the magnitude of average node mobility (in simulations, hello packets are exchanged every 5 seconds). A node moving into a new neighborhood cannot communicate with its neighbors until it has exchanged hello packets with them to learn about the fixed channels being used. Hello packet exchange is used by many routing protocols (such as AODV) as well, and link layer "Hello" information could be merged with messages from routing layers.

Changing fixed channel:

Before initiating a new Hello transmission, a node consults its *ChannelUsageList*. If the number of other nodes using the same fixed channel as itself is large, then a node with some probability p (set to 0.4 in simulations) changes its fixed channel to a less used channel. After this, the node transmits a hello packet informing neighbors of its (possibly new) fixed channel. The probabilistic approach is used to avoid frequent change of fixed channels.

We use two-hop neighborhood information in constructing *ChannelUsageList* since in IEEE 802.11 protocol, when a node A is receiving a packet from a node B on some channel i, all neighbors of B (which are two-hop neighbors of A) are required to not use channel i (this is enforced through the NAV and physical carrier sense mechanisms). Consequently, it is beneficial to ensure that the number of nodes using any given channel as their fixed channel is balanced in a two-hop neighborhood.

We do not use channel load information to switch fixed channels. Using channel load may be beneficial if the load in the network does not change frequently. On the other hand, if the load in the network changes frequently, say when there are many short-lived flows, it may lead to frequent and unnecessary channel switching. For example, HTTP transfers are often less than a second long, and if such short-lived flows dominate the network traffic, then it may lead to frequent channel switching. Basing fixed channel switching decisions on the network topology requires switching only when the topology changes, which can be of the order of tens to hundreds of seconds even with moderate mobility. Hence, we have chosen to switch fixed channels based on the number of nodes using a channel.

IV.D. Benefits of the link layer protocol

In summary, the proposed link layer protocol has the following benefits:

- A sender and a receiver do not have to synchronize before each packet transmission.
- By carefully balancing the assignment of fixed channels of different nodes over the available channels, all channels can be utilized, and the number of contending transmissions in a neighborhood significantly reduces.
- The protocol can easily scale if the number of available channels increases, and is fairly insensitive to interface switching delay.

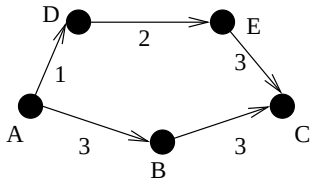

Figure 3: Need for selecting channel diverse routes

V. Multi-Channel Routing Protocol

Any existing routing protocol can potentially be used over the previously described link layer protocol (since the link layer protocol transparently manages multiple channels and interfaces). The link layer protocol ensures that within a neighborhood, different nodes use different channels to the extent possible. However, if a multi-hop route is not carefully selected, successive hops may not use different channels, thereby not fully utilizing channel diversity.

For example, popular on-demand routing protocols, such as AODV and DSR, use the shortest-path metric for route selection. Shortest-path metric assigns an unit cost for each hop in a route, and does not distinguish between a route that uses many channels, and a route that uses few channels. Figure 3 illustrates the need for choosing channel diverse routes. Suppose all nodes have already chosen their fixed channels as shown in the figure. Route A-B-C requires fewer hops, but route A-D-E-C uses different channels on each hop and can potentially support higher end-to-end throughput, even though it uses more hops. In this section, we propose a new Multi-Channel Routing metric (MCR) that selects such channel diverse paths.

WCETT [3] is a routing metric that has been proposed for selecting channel diverse paths. However, WCETT was designed for the scenario where interfaces are permanently fixed on specified channels. In our architecture, switchable interfaces have to switch among multiple channels, and the routing metric has to account for switching cost as well. The proposed MCR metric modifies WCETT to incorporate switching cost.

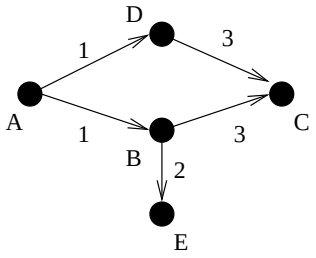

Figure 4: Need for interface switching cost

Figure 4 illustrates the need to account for the switching cost. Assume that node B is already transmitting data to node E. Suppose node A is setting up a route to node C, with two possible routes: A-B-C, and A-D-C. Both routes A-B-C, and A-D-C use the same set of channels, and hence have the same channel diversity. However, if route A-B-C is chosen, node B has to frequently switch between channels 2 and 3 when sending data to node E and node C respectively. Such frequent switching may incur a significant overhead, and the throughput over both flow A-B-C and flow B-E may reduce.

In the rest of this section, we first describe our approach to measuring switching cost at a node for each channel. We then revisit a link[1] cost metric (called ETT) used in WCETT, and highlight the modifications needed for the scenario where the number of interfaces per host is smaller than the number of channels. Individual link costs and switching costs are then combined into a new path metric (MCR) that is used by the routing protocol.

V.A. Measuring interface switching cost

The interface switching cost of a channel should measure the increase in delay a packet experiences on account of interface switching. Since a packet sent out on a fixed interface is never delayed waiting for the interface to switch, we set the switching cost of a fixed channel to be 0.

Frequent interface switching is necessitated if a node has large amounts of data to send on more than one switchable channel. For identifying such scenarios, on every switchable channel j, we measure the likelihood that the switchable interface is on a different channel when a packet has to be sent out on channel j. To estimate this, we maintain a variable *InterfaceUsage(j)* for each channel j to measure what fraction of the time a switchable interface was transmitting on channel j. If during some time interval the interface is tuned to a channel j, but is idle, then that time is not included as part of *InterfaceUsage(j)*, as the idle time could have been used to transmit data on some other channel, if it was so required. Interface usage of each channel is maintained as an exponentially weighted average over one second intervals (thus the sum of *InterfaceUsage* values over all channels is less than or equal to 1 second).

If a route chooses to use some channel j, then we estimate the probability $p_s(j)$ that the switchable interface will be on a different channel ($i \neq j$) when a packet arrives on channel j to be:

$$p_s(j) = \sum_{\forall i \neq j} \mathit{InterfaceUsage}(i) \qquad (1)$$

Note that $p_s(j)$ computation assumes that all the current interface idle time can potentially be used on channel j. The switching cost of using channel j is then measured as:

$$\mathit{SC}(j) = p_s(j) * \mathit{switchingDelay} \qquad (2)$$

where *switchingDelay* is the interface switching latency, which can be estimated offline.

Considering switching cost during route selection ensures *paths that require frequent switching are not preferred*. When some flow is already passing through a node and using some channel j, the switching cost of all other channels (except the fixed channel) will be high. Hence, new routes through the node using any other channel will have a higher cost, and therefore will not be preferred.

V.B. Measuring individual link costs

The cost of a link is measured as the expected transmission time (ETT) required to transmit a packet over the link. ETT [3] of a link (between a pair of nodes on some channel j) is defined to be:

$$ETT = ETX * \frac{S}{B} \qquad (3)$$

where ETX [22] is the expected number of transmission attempts (including retransmissions) required for transmitting a packet (equation 4, described later), S is the average packet size (which can be set to any reasonable value, say, 1024 bytes), and B is the data rate of the link. When "autorate" feature is enabled or a combination of IEEE 802.11a and IEEE 802.11b hardware is used, different links may use different data rates. The link data rate can be measured using probe packets [3], or can be read by querying the

[1] A link refers to a pair of nodes and a specific channel used for communicating between the nodes. When there are c channels, each pair of nodes can have up to c links.

driver (this support is available with newer hardware). In our simulations, we assume that link data rate is probed from the driver.

Measuring ETX: The technique used to compute ETX (expected transmission count) is based on a modification of the technique presented in [3]. The ETX of a link from a node X to a node Y on some channel j depends on the forward packet loss probability from X to Y on channel j (p_f), and the reverse packet loss probability from Y to X on channel j (p_r). Assuming 802.11 protocol is used, a data transmission is successful only when the packet is successfully acknowledged. Consequently, the probability that a transmission along the link fails is given by, $p = 1 - (1 - p_f) * (1 - p_r)$.

Once the link failure probability p is computed, the expected number of transmissions (ETX) needed before a transmission is successful is the given by,

$$ETX = \frac{1}{1-p} \tag{4}$$

In [3], explicit probe packets are broadcast by a node for measuring the loss rate. To reduce overheads, we use the "Hello" packets that are anyway broadcast by the link layer protocol as probe packets. The forward and reverse loss probabilities were measured in [3] using the following approach. A node X measures the loss rate to Y on some channel j by measuring the probe packet loss rate. Similarly, Y measures the loss rate from X on channel j as well, and informs X of the measured loss rate. Thus, both forward and reverse path probabilities between any pair of nodes can be estimated on every channel (since in [3], each node has an interface on every channel).

However, under our problem scenario, it is difficult to measure the loss rate between two nodes X and Y on *all channels*, using the above approach. Note that we assume that the number of interfaces may be smaller than the number of channels, and hence a *node will not have an interface listening on every channel*. When using the previously described link layer protocol, a node X can measure the packet loss probability from a neighbor Y *on its own fixed channel only*, as that is the only channel on which the node X is always listening and can correctly count the number of packets sent by Y. During route discovery procedure, when a node Y receives a route request packet from a node X on Y's fixed channel j, the forward loss probability from X to Y on channel j is known (based on Y's earlier measurements on channel j), but the reverse loss probability from Y to X is not known (as X may be using some other channel $k \neq j$ as the fixed channel).

Hence, for computing ETX, we make the simplifying assumption that *reverse loss probability is equal to the forward loss probability*, i.e., $p_r = p_f$, though this assumption may not always hold in practice (because of asymmetries arising out of interference and channel fading). Our simulation model incorporates fading, which can create asymmetric packet losses, and therefore the impact of the simplifying assumption has been accounted for in the simulations.

V.C. MCR: The path metric

The proposed Multi-Channel Routing (MCR) metric combines the measured link ETT and switching costs into a single path cost, using the technique proposed by [3] for WCETT metric.

The ETT cost of the i^{th} hop of a path is designated as ETT_i. The switching cost for the i^{th} hop is given by $SC(c(i))$, where c_i is the channel used on the i^{th} hop. The total ETT cost on any channel j, X_j, is defined as:

$$X_j = \sum_{\forall i, \text{such that } c_i = j} ETT_i \tag{5}$$

The MCR metric, which measures the path cost, is defined as:

$$\text{MCR} = (1-\beta) * \sum_{i=1}^{n} (ETT_i + SC(c_i)) + \beta * \max_{1 \leq j \leq c} X_j \tag{6}$$

where β is a weight between 0 and 1, n is the number of hops on the path, and total number of available channels is c.

The MCR metric is a weighted sum of two components, similar to WCETT [3]. The first component measures the sum of ETT and switching cost values along the path, and may be viewed as measuring the "resource" consumed along the path. The second component measures the cost of the "bottleneck channel" along the path. Since transmissions along different channels do not interfere, the path throughput is constrained by the throughput along the more heavily used channel. The cost of second component will be small if a channel diverse path is used. Thus, *the second component ensures that channel diverse paths are selected*, while the *first component ensures that adding more hops to a path increases its cost*. Simulation results with various β (not included for lack of space) suggest that β should not be close to either 0 or 1, and throughput is fairly insensitive to values in between. Therefore, we choose a value of 0.5 in all simulations.

In summary, MCR metric differs from WCETT in two aspects. First, MCR incorporates switching costs

into the path cost. Second, MCR uses a modified technique for computing the link loss probabilities used in link ETT computation.

V.D. Routing protocol

The MCR metric is incorporated into a source-initiated on-demand routing protocol, similar to DSR. The route discovery process is initiated by a source node, which broadcasts a Route Request (RREQ) packet over all channels[2]. Each new route discovery initiated by a node uses an unique sequence number which is included in all RREQ packets. The RREQ packet sent by a node X over a channel i contains the ETT, switching cost, and channels used on all previous hops, as well as the switching cost of channel i at node X. On receiving a RREQ, a node can compute the ETT of the previous hop based on the previously measured link loss rate. An intermediate node re-broadcasts the RREQ in the following two cases:

1. The sequence number in the RREQ is being seen for the first time. In this case, the cost of the already traversed path is stored in a local table.

2. The cost of the already discovered path in the RREQ is smaller than the cost seen in all earlier RREQs with the same sequence number, if any.

A lower cost RREQ may traverse a longer path, and reach an intermediate node after a higher cost RREQ has been forwarded. Therefore, the second condition, not present in DSR, is required to improve the probability of discovering the least cost path.

When the destination receives a RREQ, it responds with a route reply (RREP) only if the cost of the received RREQ is smaller than other RREQs (containing the same sequence number) seen till then. We use a procedure called "Route Refresh" (not present in DSR), wherein, a new route discovery is periodically initiated (the period is set to 20s in simulations) to update the costs of known routes, even if they are not broken. This mechanism ensures that the route cost information is never stale, and new lower cost routes, if any, are discovered.

The routing protocol retains other features of DSR, such as route repair using RERR messages. However, we have not used optimizations such as route caches and packet salvaging, though it may be possible to extend the protocol with those optimizations.

[2]Overhead can be reduced by initially sending requests over a subset of channels only, and later sending requests over all channels if no route was discovered during the initial attempt.

VI. Evaluation

We have simulated the proposed protocols in Qualnet version 3.6 [23]. The protocols were implemented without requiring any modifications to the IEEE 802.11 MAC layer. In all simulations, nodes in the network were equipped with two IEEE 802.11a interfaces. The duration of each simulation is 100 seconds. Unless otherwise stated, the interface switching delay is assumed to be 1 ms. We have evaluated our protocol with both CBR and FTP traffic, but only FTP results are presented here for lack of space.

The performance of our multi-channel protocols has been evaluated with 2, 5, and 12 channels (designated as "MCR - x", where x is the number of channels). Evaluation with 2 channels models the scenario with the two interfaces fixed permanently. At least 5 orthogonal channels are available with currently available 802.11a hardware, while 12 channels are provisioned for in the 802.11a standard, and therefore we simulate with these values as well. The results have been compared with a single channel network running DSR (designated as "One Channel") to quantify the benefits of using multiple channels.

VI.A. Single flow performance

We first evaluate the performance of the proposed approach in simple chain topologies. The length of a chain is varied from 1 to 10 hops. A FTP flow is setup from the first node to the last node of the chain. We set the data rate of all channels to 54 Mbps, the maximum rate possible with IEEE 802.11a. Nodes in a chain are stationary, and direct communication is possible only between adjacent nodes on the chain (distance between adjacent nodes is 40m). This scenario tests the effectiveness of the link layer protocol (routing metric is not tested here, as there is only one route between the source and the destination), and highlights the benefits of using multiple channels.

Figure 5 compares the flow throughput of a one channel network with the flow throughput of a multi-channel network using the proposed MCR metric. The FTP throughput in a single channel network rapidly degrades when the number of hops along a chain increases (this behavior is well-known) because of two reasons. First, intermediate nodes cannot simultaneously receive and forward data, cutting the achievable throughput by half. Second, since a single channel is used, transmissions on a hop will inhibit transmissions on other hops that are within the carrier sense range, thereby further degrading the achievable throughput.

When multiple channels and multiple interfaces are

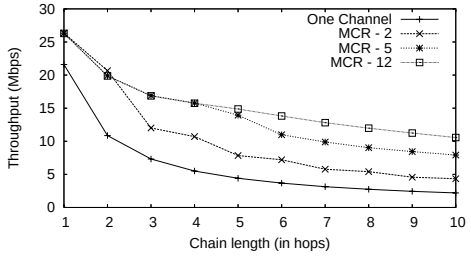

Figure 5: Performance of single FTP flow

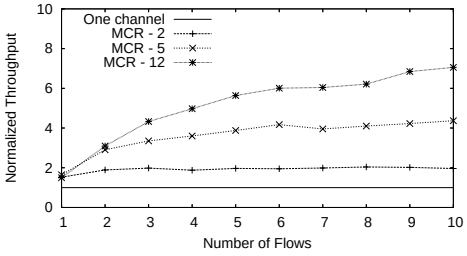

Figure 6: Impact of varying traffic load

available, the link layer protocol assigns the fixed interface of successive nodes along the chain to different channels. Also, when an intermediate node is receiving on the fixed interface, it can simultaneously forward data to the next node using the switchable interface. Consequently, MCR offers higher throughput by *using different channels on successive hops*, and by using the two interfaces to *receive and send data in parallel*.

From Figure 5 we can observe that more channels are useful with longer chains. For example, over a chain of two hops, only two channels can be utilized (one channel for each hop). Therefore, the performance with 5 and 12 channels is the same as the performance with 2 channels over a two hop chain (though higher than that of one channel). On the other hand, over a chain of 10 hops, more channels can be utilized over different hops, and therefore, having 12 channels is better than having 5 channels (and 5 channels is better than 2 channels).

To summarize, multiple channels can significantly improve flow throughput in multi-hop scenarios. Furthermore, even with only a few interfaces, it is better to use all the channels, by switching interfaces.

VI.B. Network performance

In this section, we evaluate the performance of MCR in random topologies with multiple flows. We generated 10 random topologies, and each data point (in all the remaining results) is the average over the 10 random topologies. Each random topology had 50 stationary nodes located in a 500m X 500m area. Since the aggregate throughput obtained depends on the topology, we normalized all results with the throughput obtained when using DSR on a single channel. The normalized throughput quantifies the performance improvement of multi-channel protocols with respect to a single channel network.

Figure 6 compares the throughput of MCR with a one channel network when using FTP traffic. The number of simultaneous FTP flows is varied from 1 to 10. As we can see from the figure, when the number of flows increases, MCR offers significantly better performance than a one channel network, especially when more channels are available (e.g., 12 channels offer significant improvement over using only 2 channels).

When the number of flows is small, the throughput improvement with MCR depends on the channel diversity available on the best route between the source and the destination. Therefore, the throughput improvement depends on the underlying topology. In addition, throughput improvement is larger when the length of the flow increases. Recall from the single flow result (Figure 5) that as the length of flows increase, having more channels is better (the length of flows for this experiment ranged from 1 hop to 5 hops).

When the number of flows is large, the available channel diversity can be better exploited. Furthermore, increasing the number of flows in the network increases the average contention at the MAC layer. When multiple channels are available, the fixed channels of various nodes are distributed across the available channels. Since the number of nodes using a specific channel decreases, *overheads of MAC layer contention on each channel reduces*. Therefore, by using all the available channels, MCR can provide better scalability with increasing network contention, than a single channel solution.

Although having more channels improves performance (in Figure 6, 12 channels is better than 5 channels, and so on), we can observe diminishing returns in the improvement seen (5 channels offer up to 4.2 fold improvement, while 12 channels offer only 6.9 fold improvement). The link layer protocol tries to assign different channels to different nodes in an interference neighborhood. If the number of nodes in a given neighborhood is smaller than the number of channels, then some channels are left unassigned. Consequently, for a given node density, there is some

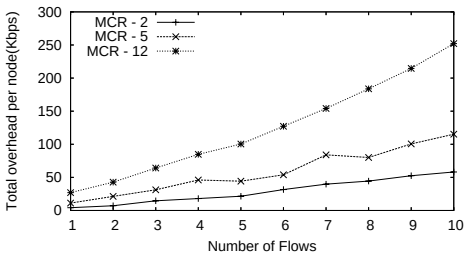

Figure 7: Total overhead of MCR, with varying traffic load

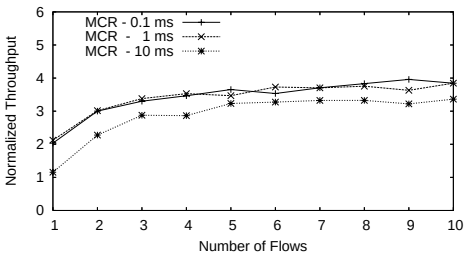

Figure 8: FTP throughput with varying switching delay

threshold number of channels beyond which adding more channels (without adding more interfaces) does not improve performance.

The magnitude of improvement reduces with increasing number of channels also because of the increased overheads. Figure 7 plots the per-node total overhead of MCR for the scenario used in Figure 6. The overhead includes the cost of sending "Hello" messages and route management (RREQ, RREP, RERR messages). As we can see from the figure, the overhead increases linearly as the number of flows increase. For each flow, overhead is incurred in discovering and maintaining a route. Since we have not incorporated caching mechanisms, there is a linear increase in the routing overhead with increasing number of flows.

The slope of the overhead curve depends on the number of channels available, with larger number of channels implying a steeper increase in the overhead. Since broadcast is supported by sending a separate packet on each channel, the overhead, for a given number of flows, increases when more channels are available. However, the *overhead per channel* remains fairly constant, as adding channels does not increase the number of broadcast packets sent on a channel. Our measurements show that overhead traffic consumes only a small fraction of the available channel capacity (even at the lowest data rate of 6 Mbps supported by 802.11a).

The key problem with increasing number of channels is that although channels are not any more congested, interfaces at nodes become congested. When more channels are available, a switchable interface has to switch to more channels and send a copy of the broadcast packet. Therefore, with more channels, a larger fraction of time is spent by the switchable interface in sending broadcast transmissions, which can limit performance.

In summary, MCR can utilize a large number of channels fairly effectively, even with only two interfaces per node. However, the magnitude of improvement seen with more channels diminishes with more channels. The number of channels that can be utilized depends on the node density and the total broadcast traffic in the network.

VI.C. Impact of switching delay

We next evaluate the impact of interface switching delay on the performance of the MCR. The interface switching latency depends on the available hardware, and current hardware imposes switching latency in the range of a few milliseconds, while future hardware may be able to achieve delays of a few hundred microseconds. Therefore, we present simulation results for switching delay varying from 0.1 milliseconds to 10 milliseconds. The results are for MCR with 5 channels.

Figure 8 plots the normalized throughput of MCR with FTP traffic. We see that the throughput degradation is minimal with moderate delay (1 ms) when compared to low delay (0.1 ms). However, with very high switching delay (10 ms), the throughput degradation is larger. Nevertheless, in all cases, MCR continues to provide significantly better throughput than in a single channel network, which implies MCR is fairly tolerant to large values of switching delay.

Higher switching delay affects performance by increasing the cost of a broadcast (since each broadcast requires switching channels), and by increasing the end-to-end delay of a flow if the best route for a flow requires switching at some node along the flow. When a route with frequent switching is used with TCP traffic, the path RTT increases. TCP throughput is inversely proportional to the RTT of the path, and therefore degrades with higher RTT. As long as the fraction of path RTT contributed by switching delay is small, switching delay has minimal impact on throughput (therefore, for 1 ms delay, there is little degradation in the throughput). When the switching

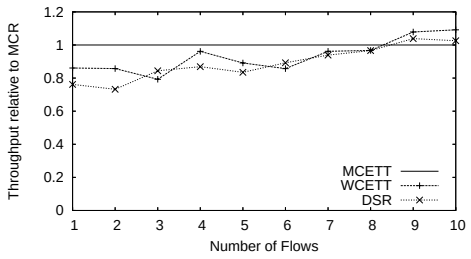

Figure 9: Comparison of proposed MCR metric with WCETT and DSR

delay starts contributing to a larger fraction of the path RTT, throughput degradation is more significant.

VI.D. Metric comparison

We next evaluate the impact of routing metrics on performance. Recall that potentially any routing protocol may operate over the proposed link layer protocol. We consider a 5 channel network (with 2 interfaces), and different routing protocols are operated over the link layer protocol.

Figure 9 compares the throughput obtained with DSR (shortest hop metric) and WCETT, with the proposed MCR metric. The throughputs are normalized with respect to MCR throughput. As we can see from the figure, when there is a single flow, DSR performs the worst (by about 23%), while WCETT is also significantly worse (by about 16%). DSR does not account for either channel diversity or switching costs, and hence does not match the performance of MCR. Although WCETT does account for channel diversity, it does not account for switching cost. Consequently, with WCETT, both forward and reverse traffic (TCP data and TCP ACKs) use the same route. MCR also initially uses the same route for both forward and reverse traffic. However, when it refreshes the routes later (see Section V.D), it accounts for any switching that may be caused at an intermediate node that handles both forward and reverse traffic. Under this scenario, MCR may select different routes for forward and reverse traffic, thereby improving performance. Similar benefits are seen even with more than one flow.

As the number of flows increase, WCETT and DSR are comparable, and start approaching MCR. With a large number of simultaneous flows (9 or 10), WCETT and DSR are marginally better than MCR. When the number of flows increase, the choice of channels chosen on any given flow is less important, as all the flows collectively will utilize most channels. Under this scenario, it becomes more appropriate to just use shortest paths. Therefore, DSR performance is close to MCR. MCR is marginally worse than WCETT with high load because switching costs included in MCR may cause longer routes to be chosen. Recall that the fixed channel has a switching cost of 0. As a result, when switching cost is high (under high load) MCR tries to choose routes that only use fixed channels, which can lead to longer routes. It is part of our future work to better estimate switching cost with large number of flows.

VII. Discussions and Future Work

In this paper, we have not considered the problem of identifying the optimal number of fixed and switchable interfaces to use, when more than two interfaces are available. In our proposal, one fixed interface is always required to allow neighbors of a node to communicate with it. However, whether multiple fixed interfaces are beneficial depends on the traffic being forwarded by a node. For example, if M interfaces are available, some K of the M interfaces can be chosen to be fixed interfaces. The fixed interfaces are mostly used for receiving data, while the switchable interfaces are mostly used for sending data. Hence, one choice for K will be to set it to approximately $M/2$ if a node receives and forwards nearly equal amounts of data. However, if a node is mostly a source (destination) of traffic, and therefore mostly transmits (receives) data, then it is better to use a smaller (larger) value of K.

It is part of our future work to develop a mechanism for dynamically changing the number of fixed interfaces at a node, based on the amount of data being transmitted and received by the node. Other avenues of future work include developing mechanisms for reducing the cost of broadcast, and studying the impact of different protocol parameters on performance. Implementation of the proposed architecture in a Linux-based testbed is in progress.

VIII. Conclusions

In this paper, we have proposed link-layer and routing protocols for multi-channel, multi-interface ad hoc wireless networks. The link-layer protocol uses the notion of *fixed* and *switchable* interfaces, and can utilize all the available channels even if the number of interfaces per host is smaller than the number of available channels. We have presented a new routing metric and incorporated the metric in an on-demand routing protocol that selects high-throughput routes in

multi-channel, multi-interface networks. Simulation results have shown that network capacity can be significantly improved by using multiple channels, even if only two interfaces per node are available.

References

[1] *IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification, P802.11*, 1999.

[2] P. Bahl, A. Adya, J. Padhye, and A. Wolman, "Reconsidering Wireless Systems with Multiple Radios," *ACM CCR*, July 2004.

[3] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *ACM Mobicom*, 2004.

[4] P. Kyasanur and N. H. Vaidya, "Capacity of Multi-Channel Wireless Networks: Impact of Number of Channels and Interfaces," in *ACM Mobicom*, 2005.

[5] P. Kyasanur and N. H. Vaidya, "Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks," in *IEEE WCNC*, 2005.

[6] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," *Ietf Manet Working Group (Draft 10)*, 2004.

[7] M. Alicherry, R. Bhatia, and L. Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *ACM Mobicom*, 2005.

[8] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *ACM Mobicom*, 2005.

[9] A. Nasipuri, J. Zhuang, and S.R. Das, "A Multichannel CSMA MAC Protocol for Multihop Wireless Networks," in *IEEE WCNC*, 1999.

[10] N. Jain, S. Das, and A. Nasipuri, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks," in *IC3N*, 2001.

[11] J. So and N. H. Vaidya, "Multi-channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals using a Single Transceiver," in *ACM Mobihoc*, 2004.

[12] M. X. Gong and S. F. Midkiff, "Distributed Channel Assignment Protocols: A Cross-Layer Approach," in *IEEE WCNC*, 2005.

[13] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks," in *IEEE Broadnets*, 2004.

[14] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," in *Ietf RFC 3561*, July 2003.

[15] N. Shacham and P. King., "Architectures and Performance of Multichannel Multihop Packet Radio Networks," *IEEE Journal on Selected Area in Communication*, vol. 5, no. 6, pp. 1013–1025, July 1987.

[16] J. So and N. H. Vaidya, "A Routing Protocol for Utilizing Multiple Channels in Multi-Hop Wireless Networks with a Single Transceiver," Tech. Rep., UIUC, Oct 2004.

[17] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *ACM Mobicom*, 2004.

[18] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *MC2R*, vol. 8, no. 2, pp. 50–65, April 2004.

[19] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *IEEE Infocom*, 2005.

[20] "Atheros inc," http://www.atheros.com.

[21] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," in *IEEE Infocom*, Hong Kong, 2004.

[22] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *ACM Mobicom*, 2003.

[23] Scalable Network Technologies, "Qualnet simulator version 3.6," http://www.scalable-networks.com.