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

Technical Report

May 2005

Pradeep Kyasanur
Dept. of Computer Science, and
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign
Email: kyasanur@uiuc.edu

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

***Abstract*— Wireless technologies, such as IEEE 802.11a, that are used for ad hoc networks provide for multiple non-overlapping channels. Most ad hoc routing protocols that are currently available are designed to use a single channel. The available network capacity can be increased by using multiple channels, but this requires the development of new protocols specifically designed for multi-channel operation. This paper presents protocols for improving the capacity of multi-channel wireless networks. Our protocols simplify the use of multiple channels by using multiple interfaces, although the number of interfaces per node 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 routing protocol that operates over the link layer protocol, and is specifically designed for multi-channel, multi-interface ad hoc wireless networks. 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 is smaller than the number of channels.**

***Index Terms*— Ad hoc networks, routing, multiple channel, multiple interfaces**

I. Introduction

Wireless technologies, such as IEEE 802.11a [1], provide for multiple non-overlapping channels[1]. Multiple channels have been utilized in infrastructure-based networks by assigning different channels to adjacent access points, thereby minimizing interference between access points. However, typical multi-hop wireless network configurations have used a single channel to ensure any adjacent pair of nodes can communicate. For meeting the ever-increasing throughput demands of applications, it is necessary 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], thereby not using the remaining $c - m$ channels. However, such an approach may offer only a m-fold (potentially $m \ll c$) increase in the network capacity. Our goal in this work is to utilize all the available channels and achieve close to a c-fold increase in capacity, 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.

For addressing the challenges that arise when trying to utilize all the available channels with a few interfaces, we have proposed a new architecture [5] for multi-interface networks. In our architecture, the available interfaces are classified into "fixed" and "switchable" interfaces. Fixed interfaces stay on specified "fixed channels" (can be different for different 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.*

[1]Some of the non-overlapping channels may interfere with each other (see Section III-B)

In this paper, we present a new link-layer protocol that can be used to implement the notion of fixed and switchable interfaces. The link-layer protocol can be built over existing commodity 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*. The MCR metric builds upon an existing multi-channel metric called WCETT [3]. WCETT was designed under the assumption number of interfaces per node is equal to number of channels, and requires many changes (that we have incorporated) before it can be used under our problem setting. We evaluate our proposed MCR metric with an on-demand source routing protocol (similar to DSR [6]).

To summarize, our contributions in this paper are:

- We present a multi-interface solution for exploiting multiple channels. The solution can be implemented with commodity IEEE 802.11 hardware.
- We propose a new link-layer protocol that simplifies coordination among nodes, while utilizing multiple channels, and is well-suited for ad hoc networks.
- We propose a new routing metric that accounts for channel diversity and interface switching cost.

Our results show that, for example, even with only two interfaces, a five channel network can offer more than five-fold improvement (we explain later why greater than five-fold improvement is possible) over a single channel network, while maintaining network connectivity.

The rest of the paper is organized as follows. We describe related work in Section II, and define 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

Several researchers have proposed MAC protocols for utilizing multiple channels [7]–[12]. These multi-channel protocols require changes to existing standards, such as IEEE 802.11, and therefore cannot be deployed by using commodity hardware. In contrast, our proposal can be implemented with standard 802.11 interfaces.

Adya et al. [13] propose a link-layer solution for striping data over multiple interfaces, but their solution is designed for the scenario where number of interfaces is equal to number of channels. Bahl et al. [14] propose SSCH, a link layer solution that uses a *single interface*, and can run over unmodified IEEE 802.11 MAC. In contrast, our link layer protocol is designed for multi-interface networks, and we complement the link layer protocol with a routing protocol as well.

Existing routing protocols for multi-hop networks such as DSR [6] and AODV [15] 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. [16] have proposed an architecture for multi-channel wireless networks that uses a single interface. So et al. [17] have proposed a routing protocol for multi-channel networks that uses a single interface at each node. We propose to use multiple interfaces, which can offer better performance than a single interface solution. Furthermore, our solution retains the connectivity present in a single channel network.

There are a few routing proposals specific to 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 *static* mesh networks with the assumption that all traffic is directed toward specific gateway nodes. In contrast, our proposal is designed for more general *mobile* ad hoc networks, where potentially any node can 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 have developed a new MCR metric by incorporating suitable modifications to WCETT.

We have studied the theoretical benefits of using multiple channels and multiple interfaces in [4]. In an earlier paper [5], we have also presented a classification of interface assignment strategies, and proposed a new architecture that classifies interfaces into fixed and switchable interfaces. In this paper, we present a link layer protocol to implement the proposed architecture, and a new routing protocol as well.

III. Preliminaries

A. Terminology

We define an "interface" to be a network interface card equipped with a half-duplex radio transceiver, for example, a commodity IEEE 802.11 wireless card. A "channel" is a part of the wireless spectrum with a specified bandwidth. For example, a channel in IEEE 802.11a standard has a bandwidth of 20 MHz. Two channels are defined to be "orthogonal" (or non-interfering), if they use non-overlapping parts of the spectrum *and* do not interfere with each other.

B. Number of orthogonal channels

IEEE 802.11a standard allocates 12 non-overlapping channels. 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. Future changes in FCC regulations may provide for more orthogonal channels as well.

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.

C. Interface switching latency

When the number of interfaces is smaller than the number of available channels, periodically switching an interface from one channel to another may enable utilization of all the available channels. However, 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 the range of a few hundred microseconds [21] to a few milliseconds [22]. It is expected that with improving technology, the switching delay can be reduced to a few tens of microseconds [14]. Protocols that utilize interface switching need to be flexible enough to accommodate a range of switching delays. Most of our simulation results use a conservative estimate of 1 ms for switching delay, and we have evaluated the protocol for other values of switching delay as well.

D. Assumptions and Problem Formulation

The protocols proposed in this paper are designed for a *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 define the requirements 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 is smaller than the number of available channels*. The solution must be flexible enough to accommodate different number of channels and interfaces.
2) Ensure that a network which is connected when using a single common channel, continues to be connected when multiple channels are being used.

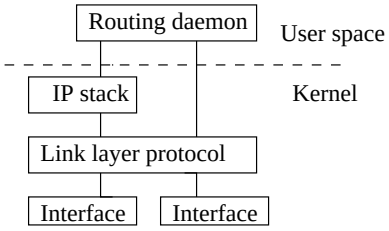

Fig. 1. Proposed architecture

3) Allow implementation on existing 802.11 hardware.

We next discuss our solution approach, followed by detailed protocol descriptions.

E. Solution approach

We develop a link layer protocol to manage the use of multiple interfaces, and a routing protocol 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. Figure 1 outlines the proposed architecture. A further benefit of this approach is that even existing routing protocols can be used without any modifications, since our link layer protocol completely hides the complexity of managing multiple channels and interfaces from the higher layers. We are currently implementing this architecture in a Linux-based testbed.

In Section IV we will describe the link layer protocol. Section V presents a new routing metric, MCR, designed for multi-channel operation, and an on-demand routing protocol that uses the new metric.

IV. Link layer protocol

When the number of available interfaces is smaller than the number of available channels, an interface assignment strategy is required to assign interfaces to specific 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 the channel being used by at least one interface of X. We have provided a classification of interface assignment strategies in [5]. In [5], we have also presented an interface assignment architecture, which is briefly described in the next sub-section. We then describe the new link layer protocol proposed in this paper for implementing the architecture.

A. Background: Interface assignment architecture [5]

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

- *Fixed Interfaces:* Some K of the M interfaces at each node are assigned for long intervals of time to some

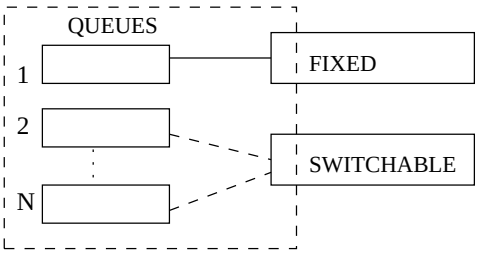

Fig. 2. Illustration of queues associated with N channels and 2 interfaces

K channels, and we designate these interfaces as "fixed interfaces', and the corresponding channels as "fixed channels".

- *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 channel to which a switchable interface is assigned to is called a "switchable channel".

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 X. In the rest of this section, we describe the new link layer protocol for implementing this architecture, and explain how the protocol can utilize all the available channels.

B. Managing communication between nodes

For simplifying the description of the protocol, we assume $M = 2, K = 1$ for all nodes, i.e., there is one fixed, and one switchable interface (although the protocol proposed next is applicable with any values of $M \geq 2$ and $1 \leq K \leq M$). 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).

1) Inserting packets into channel queues: Each channel is associated with a packet queue, as shown in Figure 2. 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 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 case 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*, 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.

2) 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 in the following two cases:

1) The switchable interface is on a channel with an empty queue, and there is at least one other non-empty queue.
2) The switchable interface has been on a channel for more than *MaxSwitchTime* duration, and there is at least one other non-empty queue. 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 5 ms (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.

3) Example of protocol operation: Figure 3 illustrates the protocol used for communication between nodes when using "fixed" and "switchable interfaces". 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 respectively. Assume that initially node A has its switchable interface on channel 3, node B has its switchable interface on channel 1, and node C has its switchable interface on channel 2. 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 are correctly set up during a flow initiation, there is no need to switch the interfaces for subsequent packets of the flow (unless a switchable interface has to switch to another channel for sending packets of a different flow).

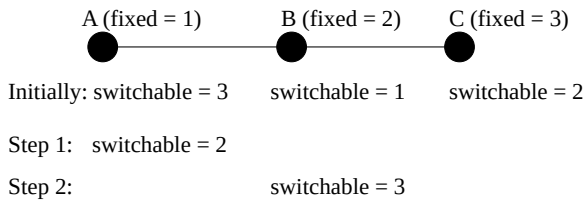

Fig. 3. Example of link layer protocol operation with 3 channels, 2 interfaces

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.

1) 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 the 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.

The frequency of hello packet exchange depends on the magnitude of average node mobility. 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 by neighbors. Hello packet exchange is used by many routing protocols (such as AODV) as well, and with moderate degrees of mobility, the overhead of hello packet exchange is not expected to be large.

2) 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 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 while the packet is being received (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 the 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.

D. Properties 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 for channel switching. Thus, the protocol is designed to *not require* a coordination algorithm for ensuring the sender and receiver are on the same channel.
- By carefully balancing the assignment of fixed interfaces 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.

V. Multi-Channel Routing Protocol

In this section, we describe the proposed Multi-Channel Routing (MCR) metric that is incorporated in an on-demand multi-channel routing protocol. The routing protocol operates over the proposed link layer 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). However, popular on-demand routing protocols such as AODV and DSR use the shortest-path metric for route selection, which does not suffice in a multi-channel network. 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 4 illustrates the need for choosing channel diverse routes. 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. The proposed MCR metric assigns lower cost to channel diverse paths (Section V-C).

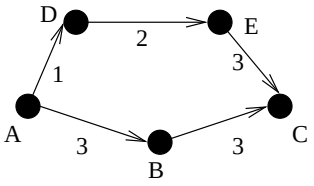

Fig. 4. Need for selecting channel diverse routes

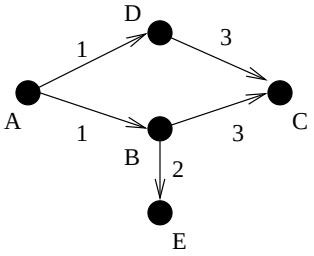

Fig. 5. Need for considering interface switching cost

Since interfaces have to be switched between channels in our problem scenario, interface switching cost has to considered during route selection as well. The expected time it takes a packet to traverse a link depends on whether interface switching is required before packet transmission or not. Figure 5 illustrates the need to account for the switching cost. Assume 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 flows A-B-C and B-E reduces, because the switchable interface on B becomes a bottleneck to performance.

In the rest of this section, we first describe our approach to measuring switching cost at a node for each channel. The link layer protocol is extended to compute the switching cost, and this cost information is used by the routing protocol. We then develop a link[2] cost metric (called ETT) that accounts for switching cost and link quality. Individual link costs are combined into a new path metric (called MCR) that is used by the routing protocol.

A. Measuring interface switching cost for each channel

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:

$$p_s(j) = \sum_{\forall i \neq j} \textit{InterfaceUsage}(i) \quad (1)$$

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 interface j is then measured as:

$$\textit{SwitchingCost}(j) = p_s(j) * \textit{switchingDelay} \quad (2)$$

where *switchingDelay* is the interface switching latency, which can be estimated from offline measurements (in most of our simulations, we set it to 1 ms).

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.

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. We define ETT of a link (between a pair of nodes on some channel j) to be:

$$ETT = ETX * \frac{S}{B} + \textit{SwitchingCost}(j) \quad (3)$$

where ETX is the expected number of transmission attempts (including retransmissions) required for transmitting a packet (equation 5, described later), *SwitchingCost(j)* is the interface switching cost on channel j (equation 2), 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, 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 driver (this support is available with newer hardware). In our simulations, we assume link data rate is probed from the driver.

In [3] ETT computation included only the first component of equation 3, but in our problem scenario it is important to include switching cost as well.

[2]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, though all links cannot be simultaneously used.

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:

$$p = 1 - (1 - p_f) * (1 - p_r) \tag{4}$$

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

$$ETX = \frac{1}{1-p} \tag{5}$$

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 fewer 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 asymmetry 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.

Note that the above loss estimation (even in the approach used by [3]) implicitly assumes that the link loss probability measured using (possibly small) broadcast packets are representative of the loss probabilities seen by (possibly large) unicast packets. Although, this assumption may not always hold in practice, the approach is still effective in *discriminating between good and bad links*.

C. MCR: The path metric

The proposed Multi-Channel Routing (MCR) metric combines the measured link ETT 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 total ETT cost on any channel j, X_j, is defined as:

$$X_j = \sum_{\forall i, \text{such that } i^{th} \text{ hop uses channel j}} ETT_i \tag{6}$$

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

$$\text{MCR} = (1 - \beta) * \sum_{i=1}^{n} ETT_i + \beta * \max_{1 \leq j \leq c} X_j \tag{7}$$

where β is a weight between 0 and 1, and 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 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*. Experiments in [3] for WCETT show that 0.5 is a reasonable choice for β, and we also use this value for MCR metric in our simulations.

D. Route discovery and route maintenance

The proposed multi-channel routing protocol is designed to be a source-initiated on-demand routing protocol, similar to DSR. The routing protocol uses the MCR metric described above. The route discovery process is initiated by a source node, which broadcasts a Route Request (RREQ) packet over all channels[3]. 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 both the ETT and channels used on all previous hops, as well as the switching cost of channel i at node X. A node on receiving a RREQ can compute the ETT of the previous hop based on the previously measured link loss rate, and the switching cost included in the RREQ. 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.

[3]Request 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.

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 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. This ensures that high cost paths are not unnecessarily sent back to the source node. The source node always uses the least cost route received from the destination for routing data packets.

We use a procedure called "Route Refresh", 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.

VI. Evaluation

We have simulated the proposed protocols in Qualnet version 3.6 [23]. We added a layer above the MAC to implement the link layer protocol, and the routing protocol was implemented at the network layer. No modifications were required 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 orthogonal channels (designated as "MCR - x", where x is the number of channels). The results have been compared with a single channel network running DSR (designated as "DSR - 1") to quantify the benefits of using multiple channels. The results with 2 channels (recall that we are using two interfaces) quantifies the performance of WCETT metric [3], as in this scenario, the number of channels is equal to number of interfaces, resulting in same routes being chosen by both WCETT and our proposed MCR metric. 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.

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.

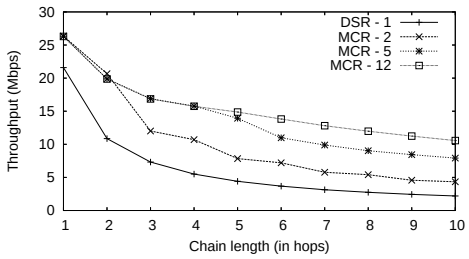

Fig. 6. Performance of single FTP flow

Figure 6 compares the flow throughput with DSR (in a 1 channel network), and the flow throughput with the proposed MCR metric (in multi-channel networks). The FTP throughput in single channel networks 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 other transmissions on other hops that are within the carrier sense range, thereby further degrading the achievable throughput.

When multiple channels and multiple interfaces are used in MCR, the link layer protocol assigns the fixed channel 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 6 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).

The key observation from Figure 6 is that *multiple channels can significantly improve throughput of a flow* in multi-hop scenarios. Furthermore, even with only a few interfaces (2 in this example), having large number of channels (up to 12 channels in this example) is beneficial. Next, we study the benefits of multiple channels with multiple flows.

B. Network performance

In this section, we evaluate the performance of MCR in random topologies with multiple flows. 50 nodes are initially

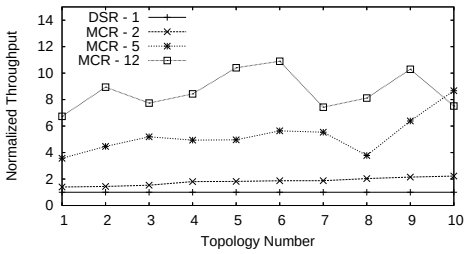

Fig. 7. FTP throughput in random topologies

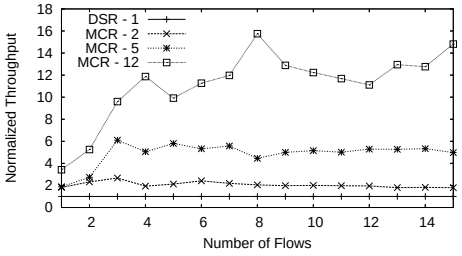

Fig. 8. Impact of varying traffic load

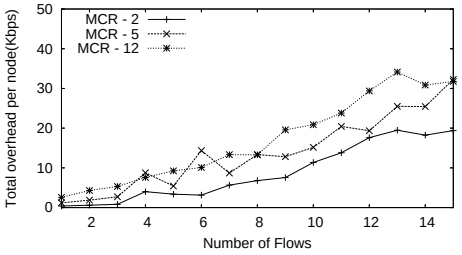

Fig. 9. Total overhead of MCR, with varying traffic load

placed in a 500m X 500m area, and nodes move using the random way-point mobility model, with both the maximum and the minimum speeds[4] set to 10m/s. "Autorate" was enabled on each interface (as is commonly done in practice). Since the aggregate throughput obtained depends on the topology, we normalize all results with the throughput obtained when using DSR on a single channel. The normalized throughput quantifies the performance improvement of multi-channel MCR with respect to a single channel network.

Figure 7 compares the throughput of single channel DSR with multi-channel MCR when using FTP traffic. 5 FTP flows are setup between randomly selected pair of nodes. Results are plotted for 10 different random topologies. The topologies are numbered from 1 to 10 in the increasing order of normalized throughput with 2 channels ("MCR - 2" curve).

The throughput of MCR with 2 channels varies from 1.4 times of DSR to 2.2 times of DSR depending on the topology. Similarly, for MCR with 5 channels, the throughput varies from 3.5 times to 8.6 times that of DSR, and for MCR with 12 channels, the throughput varies from 6.7 times to 10.9 times that of DSR. The results suggest that *even if only two interfaces are available, more than two channels can be effectively utilized*. In many scenarios, MCR with k channels can offer more than a k fold improvement in performance because distributing load over multiple channels reduces contention and collision overheads.

The improvement obtained with MCR strongly depends on the underlying topology. If multiple routes are available between a pair of nodes, then MCR will choose a good *channel diverse* route, resulting in significant throughput improvement. Otherwise, MCR may be forced to use a less channel diverse route, resulting in lower throughput improvement. In general, the results suggest that MCR *can offer significant improvements even when using only two interfaces*. Furthermore, the simulations involve moderate mobility, which indicates the suitability of MCR protocol in mobile ad hoc networks.

We also evaluate the impact of varying the number of flows in the network. We choose the random topology 5 from Figure 7 (which is the topology with median performance). The number of FTP flows in the network is varied between 1 to 15. Figure 8 compares the throughput of MCR with DSR. As the number of flows increases, MCR offers significantly better performance than DSR, especially when more channels are available. For example, MCR with 12 channels offers 3.4 times the throughput of DSR with 1 flow, but offers around 14.8 times DSR throughput with 15 flows.

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. Since the simulations involve mobility, the best route at different periods of time may offer different degrees of channel diversity. As a result, the full benefits of using a large number of channels (say, 5) is not realized when the number of flows is small. When the number of flows is large, at any point of time, at least a subset of flows can utilize the available channel diversity. 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, *MAC layer contention on each channel reduces* (by a factor larger than the number of channels). Therefore, by using all the available channels MCR can provide better scalability with increasing network contention, than a single channel solution.

Figure 9 plots the per-node total overhead of MCR for the scenario used in Figure 8. The overhead includes the cost of sending "Hello" messages and route management (RREQ, RREP, RERR messages). Since broadcast is supported by sending a separate packet on each channel, the overhead increases when more channels are available. Nevertheless, the *overhead per channel* remains small, and consumes only a small fraction of the available channel capacity (even at the lowest data rate 802.11a supports 6 Mbps).

Another observation from Figure 9 is that the *overhead increases linearly* as the number of flows increase. For each flow, overhead is incurred in discovering and maintaining a route.

[4]Random way-point based simulations have been shown to not stabilize [24] when the maximum and minimum speeds are widely different.

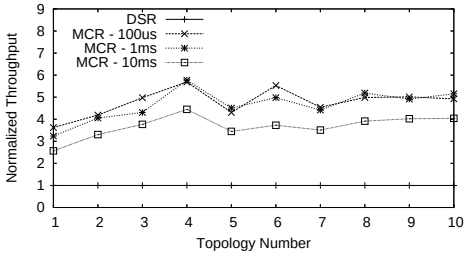

Fig. 10. FTP throughput with varying switching delay

Since we have not incorporated caching mechanisms, there is a linear increase in the routing overhead with increasing number of flows. Using caching in multi-channel networks may result in sub-optimal routes being chosen. It is part of our future work to study the impact caching (possibly sub-optimal) routes will have on protocol performance. Also, broadcast cost can be reduced by using partial broadcasts (discussed in Section IV-B.1).

C. Impact of interface switching latency

In this section, we evaluate the impact of interface switching delay on the performance of the MCR. The evaluation is over the 10 random topologies used in Figure 7. The impact of switching delay is more evident when there are more contending flows. Hence, we set up 15 FTP flows between randomly selected pair of nodes. The interface switching latency depends on the available hardware, and current estimates for switching latency range from a few hundred microseconds to a few milliseconds. Therefore, simulation results are for switching delay varying from 100 microseconds to 10 milliseconds. The results are for MCR with 5 channels.

Figure 10 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 (100 μs). However, with very high switching delay (10 ms), the throughput degradation is more significant. 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 throughput). When the switching delay starts contributing to a larger fraction of the path RTT, there is a more pronounced impact on throughput.

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 architecture, 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 when the number of interfaces 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 when only two interfaces per node are available.

IX. Acknowledgment

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

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," Tech. Rep., University of Illinois at Urbana-Champaign, March 2005.

[5] P. Kyasanur and N. H. Vaidya, "Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks," in *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. A. Marsan and F. Neri, "A Simulation Study of Delay in Multichannel CSMA/CD Protocols," *IEEE Transactions on Communications*, vol. 39, no. 11, pp. 1590–1603, November 1991.

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

[9] A. Nasipuri and S.R. Das, "Multichannel CSMA with Signal Power-based Channel Selection for Multihop Wireless Networks," in *VTC*, Sept 2000.

[10] N. Jain, S. Das, and A. Nasipuri, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks," in *IC3N*, October 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 *Mobihoc*, 2004.
[12] M. X. Gong and S. F. Midkiff, "Distributed Channel Assignment Protocols: A Cross-Layer Approach," in *IEEE Wireless Communications and Networking Conference (WCNC)*, March 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 International Conference on Broadband Networks (Broadnets)*, 2004.
[14] 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.
[15] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," in *Ietf RFC 3561*, July 2003.
[16] N. Shacham and P. King., "Architectures and Performance of Multi-channel Multihop Packet Radio Networks," *IEEE Journal on Selected Area in Communications*, vol. 5, no. 6, pp. 1013– 1025, July 1987.
[17] J. So and N. H. Vaidya, "A Routing Protocol for Utilizing Multiple Channels in Multi-Hop Wireless Networks with a Single Transceiver," Tech. Rep., University of Illinois at Urbana-Champaign, October 2004.
[18] A. Raniwala, K. Gopalan, and T. 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.
[19] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *Infocom*, 2005.
[20] "Atheros inc," http://www.atheros.com.
[21] "Maxim 2.4 GHz 802.11b Zero-IF Transceivers," http://pdfserv.maxim-ic.com/en/ds/MAX2820-MAX2821.pdf.
[22] 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, March 2004.
[23] Scalable Network Technologies, "Qualnet simulator version 3.6," http://www.scalable-networks.com.
[24] J. Yoon, M. Liu, and B. Noble, "Random Waypoint considered Harmful," in *INFOCOM*, 2003.