DCMA: A Label Switching MAC for efficient packet forwarding in multi-hop wireless networks

Arup Acharya, *Member, IEEE,* Sachin Ganu, *Member, IEEE,* and Archan Misra, *Member, IEEE*

***Abstract*— This paper addresses the problem of efficient packet forwarding in a multi-hop, wireless "mesh" network[1] . We present an efficient interface contained forwarding (ICF) architecture for a "wireless router", i.e. a forwarding node with a single wireless NIC in a multi-hop wireless network that allows a packet to be forwarded entirely within the network interface card of the forwarding node without requiring per-packet intervention by the node's CPU. To effectively forward packets in a pipelined fashion without incurring the 802.11-related overheads of multiple independent channel accesses, we specify a slightly modified version of the 802.11 MAC, called Data Driven Cut-through Multiple Access (DCMA) that uses MPLS-like labels in the control packets, in conjunction with a combined ACK/RTS packet, to reduce 802.11 channel access latencies. Our proposed technique can be used in combination with "frame bursting" as specified by the IEEE 802.11e standard to provide an end-to-end cut-through channel access. Using extensive simulations, we compare the performance of DCMA with 802.11 DCF MAC with respect to throughput and latency and suggest a suitable operating region to get maximum benefits using our mechanism as compared to 802.11**

***Index Terms*— Mesh networks, pipelined, cut-through, 802.11 MAC**

I. Introduction

WITH the reduction in prices of commodity 802.11 products and their ready availability, there has been a substantial increase in the number of users that use wireless access for information. More recently, there has been a significant effort in using 802.11 in urban metropolitan areas to provide "hot-spot" high speed coverage as well as community wireless networks [1][2]. The IEEE 802.11s Task Group [3] is also currently involved in efforts to standardize protocols for wireless mesh networks that will enable "instant wireless networks" such as in-building wireless networks in malls, hotels and apartment blocks, and community networks (where rooftop antennas are used to create an ad-hoc wireless access infrastructure in specific residential areas). If the raw channel capacity of the basic 802.11 channel could be effectively utilized, multi-hop wireless networks can indeed become a compelling low-cost broadband access alternative, especially in relatively dense urban areas [4]. However, the throughput achieved by current implementations of multi-hop 802.11 networks is still significantly lower than the underlying channel capacity. Some of the reasons for this overall lower throughput include the unfairness of the exponential backoff process on hidden nodes contending for the channel [5], the poor spatial reuse due to channel sensing-induced backoffs in the extended neighborhood of an ongoing transmission [6], the potential contention among packets of the same flow at different links [7], and the lack of an appropriate MAC layer to exploit the total number of 802.11 available channels [8]. A variety of proposals to remedy these problems have been suggested in literature, and these issues continue to be the subject of active investigation.

In this paper, however, we focus on an entirely different aspect of performance degradation in multi-hop wireless networks, namely the packet forwarding inefficiency. Our work is motivated by the observation that, in a multi-hop wireless network, the action of packet forwarding undertaken by an intermediate node is significantly different from the corresponding operations performed in a wired network. In a wired network, a router typically has at least two physical network interfaces, with the forwarding functionality consisting of receiving a packet over one physical interface and subsequently sending it out over a second interface[2]. In contrast, in a wireless network, a node N with a single wireless interface may act as an intermediary for two nodes that are each within the communication range of N but not directly within the range of each other. Thus, we see that packet forwarding in the wireless environment does not typically imply the transfer of a packet between distinct interfaces on a single host.

However, all implementations of 802.11-based packet forwarding operate at the network layer, treating the process of receiving a packet from the upstream node and of sending it to the downstream nodes as two independent channel access attempts. Figure 1 shows a conventional implementation of software-based packet forwarding. This approach involves the reception of a packet on the wireless interface, transfer of the packet up the host's protocol stack to the IP layer where a routing lookup is used to determine the IP (and MAC) address of the next hop, and subsequent transmission of the packet using the same wireless interface to the MAC address of the next hop. This mechanism introduces two forms of latency in the multi-hop wireless forwarding process that are independent of all the other 802.11-related drawbacks enumerated earlier: 1) latency related to independent channel accesses required for each hop along the path and 2) latencies associated with interrupt-handling, packet copying and route lookup at each

Manuscript received October 1, 2005; revised March 21, 2006.

A. Acharya and A. Misra are with the IBM T.J. Watson Research Center, Hawthorne, NY.

S. Ganu is a Ph.D. candidate at WINLAB, Rutgers University, Piscataway, NJ

[1]A preliminary version of this paper appeared in ACM International Workshop on Wireless and Mobile Multimedia (WoWMoM), 2002

[2]In high-end routers/switches, a packet is transferred from one interface to another via dedicated switching fabric, while in software-based routers, an incoming packet is processed by the CPU before subsequent transmission on the outgoing interface.

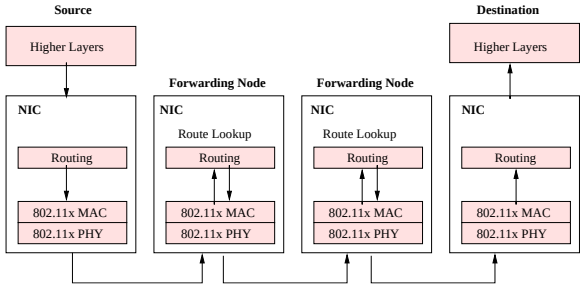

Fig. 1. Typical packet forwarding in a multihop wireless network

forwarding node.

Our primary goal in this paper is to define a practical architecture and protocol for pipelined network interface card (NIC) contained forwarding and to evaluate its effectiveness compared to basic 802.11. To this end, we first propose an architecture for a forwarding node in a multi-hop wireless network that shifts the packet forwarding functionality away from the host processor to the wireless network interface card (NIC). This is done by combining medium access control (MAC) for packet reception and subsequent transmission with address lookup in the interface card itself, using fixed-length addressing labels in the MAC control packets. This efficient "layer-2 forwarding", called Interface-Contained Forwarding (ICF) can be enabled by using label-switching on the MAC-layer data path. The information needed by the NIC to perform NIC-resident lookups can be established offline using a separate label-distribution algorithm. The choice of the actual mechanism for label distribution does not affect the performance of the ICF architecture. A label itself can be similar to an MPLS label [9]. For label distribution, existing routing protocols such as AODV [10] may be adapted or distribution mechanisms such as LDP [11] may be used. This allows packet forwarding to be confined entirely to the NIC, which matches the label of an incoming packet with an entry in a data structure to determine the MAC address of the next hop node and the label to be used for that hop.

To reap the full benefits of the optimized forwarding process, it is also necessary to define an efficient medium access protocol for packet forwarding, i.e., define an atomic channel access scheme that pipelines the reception of a packet from an upstream node and the subsequent transmission to the downstream node, to avoid the overhead of separate channel accesses on the upstream and downstream links. We present a simple modification of the 802.11 contention resolution scheme, called Data-driven Cut-through Multiple Access (DCMA) that provides preferential access to the pipelined forwarding, using a modified ACK/RTS control packet. Both of these enhancements work in tandem: to exploit the "cut-through" capability of DCMA, the NIC must be capable of determining the identity of the next-hop node from the signaling information in the contention resolution phase (without transferring the packet from the NIC to the host CPU and invoking a routing table lookup).

The rest of this paper is organized as follows. Section 2 presents a brief overview of the basic 802.11 forwarding process. Section 3 introduces the notion of label switching and its application to a multi-hop wireless network using the standard 802.11 MAC. Section 4 then presents the basic DCMA protocol, based on a simple modification to the 802.11 MAC to enable atomic packet forwarding in wireless networks. Section 5 then describes our implementation of DCMA with the ns-2 simulator and presents simulation results comparing DCMA performance with basic 802.11-based packet forwarding. This section also explains how the forwarding behavior can be modified to ensure that DCMA causes no additional unfairness in channel access over 802.11, and suggests a suitable operating region to get maximum benefits using the above mechanism as compared to 802.11. Finally, Section 6 concludes with a discussion and summary of future work and open research issues.

A. Related Work

The use of MPLS (or labels) for providing fast and efficient packet forwarding on the wireless channel has not been extensively reported in literature. In [12], the authors have suggested using MPLS to support packet routing and handoff in wireless cellular networks along with the use of label merging to accommodate multiple links between a mobile node and the cellular infrastructure. To the best of our knowledge, there appears to be no prior public work in the area of devising MAC algorithms for providing label-based forwarding in multi-hop wireless networks. DCMA's pipelined mode of channel access, described in Section 4, is similar to the contention free "burst" mode of 802.11e[13]. While the contention free burst is on a per-hop basis, DCMA extends this burst by providing an end-to-end cut-through access across multiple hops along the path. Several approaches for pipelining data transmissions have been proposed in [14][15]. More recently, in [16], the authors propose a Control Channel Based MAC (C^2M) which uses a combination of advanced channel reservation and packet aggregation on the low data rate control channel to improve the efficiency of the data channel. However, these approaches are based on using out-of-band signaling for coordinating data transmissions on the main channel, whereas our approach uses does not need a separate control channel. Also, based on our earlier work [17], alternative mechanisms for distributing labels amongst nodes have been considered such as [18][19]. Additionally, in [20], the authors propose a queue driven cut-through model that performs cut-through access at an intermediate node only for packets buffered in its queue irrespective of the flow to which they belong. Thus, cut-through is enabled only when the buffer at the intermediate relays builds up.

II. Conventional Packet Forwarding in 802.11 Networks

In this section, we briefly explain the 802.11 Distributed Coordination Function (DCF) contention resolution mechanisms commonly employed in multi-hop ad-hoc networks. Each node essentially acts a peer to all nodes within its transmission range. To avoid the hidden node problem, unicast communication in the DCF mode involves a 4-way handshake

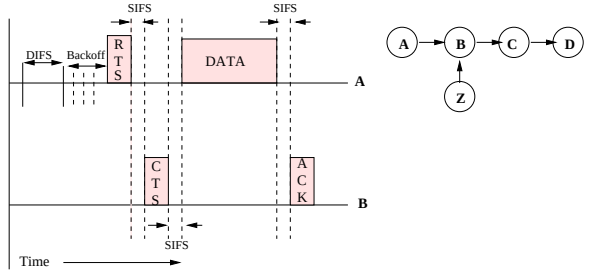

Fig. 2. 802.11 DCF Channel Access Mechanism

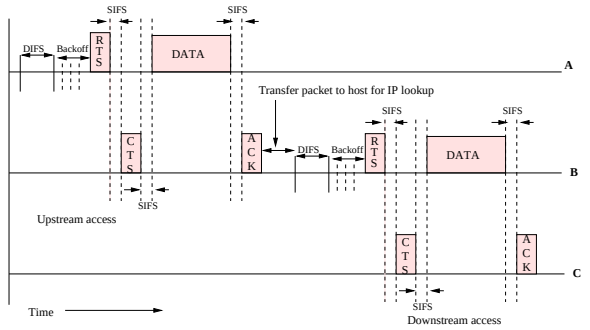

Fig. 3. Multihop forwarding in 802.11 based network

mechanism (shown in Fig. 2) between sender A and recipient B using RTS/CTS exchange prior to data transmission. The same topology represented in the Fig.2 is used for later discussions in this section.

The interaction consists of an RTS-CTS exchange that silences the neighbors in the vicinity of the sender (A) and receiver (B) respectively, followed by the data transfer and an acknowledgement. For contention resolution, 802.11 uses a timer-based exponential back-off scheme where the node selects a random back-off time in the range [0, Contention Window] (specified in terms of slots) if the channel is busy. Each time the medium becomes idle, the station waits for a DIFS and then decrements the backoff timer in units of aSlotTime. The node makes a fresh attempt at sending an RTS packet upon the expiration of the timer. Upon failure of the RTS packet, the contention window is doubled and a random timer is chosen from the new window. Each 802.11 node also maintains a Network Allocation Vector (NAV) that monitors the state of the channel. Whenever the node overhears a control packet (RTS or CTS) transmitted by a neighboring node (to some other node), it updates its NAV appropriately to reflect the duration of the corresponding 4-way data exchange. Even in the case of the 802.11a and 802.11g, the basic carrier sensing and channel access mechanism remains the same.

A. Forwarding Operation in 802.11 Ad-hoc Networks

We now discuss the overheads associated with a forwarding operation when using the 802.11 MAC in a multi-hop wireless environment. The upstream node (node A) sends a data packet to the forwarding node (node B); which then forwards the packet to the downstream node (node C), as shown in Fig. 3.

After the IP lookup function, host A determines that B is the next hop of the DATA packet, and the packet is transferred to A's NIC. The MAC implementation on A's NIC then performs a 4-way handshake (including any backoff timer-based countdown that may be needed to gain access to the channel) to forward the packet to B's NIC. At B, the packet is transferred from the device to the main memory either using DMA or PIO (Programmed I/O) techniques, and the host CPU is notified (e.g. via interrupts or soft IRQs) [21]) for further processing of the packet. The host software (IP protocol stack) would typically queue up the packet in a transmission queue and select packets for transmission based on a scheduling algorithm (typically, FIFO). When this packet reaches the head of the queue, the same steps as those executed at A, would be taken, e.g. perform lookups to determine the IP address and then the MAC address of the next hop (C), insert the MAC-layer header (corresponding to next hop C) and transfer the packet to the NIC[3]. This packet is now treated as an independent data transfer between the nodes B and C; accordingly, B performs the usual backoff timer countdown before initiating an RTS-CTS-DATA-ACK exchange with C. Once this handshake is successfully completed, the packet is received by C's NIC, at which point the whole forwarding process is repeated. As with the initial data transfer (from A to B), the NAV of node A is blocked (by the RTS sent by B) for the entire duration of the 4-way exchange between B and C.

III. The Label-based ICF Architecture

In this section, we describe the label-based Interface Contained Forwarding (ICF) architecture that reduces the latency associated with the NIC-host-NIC interaction at the forwarding node. More importantly, eliminating the NIC-host-NIC interaction enables us to subsequently perform atomic packet-forwarding at the MAC layer, eliminating the more significant latency component associated with independent channel accesses in 802.11. To support label-based forwarding, the network interface card (NIC) is enhanced to store a label-switching table, consisting of an incoming MAC address, an incoming label, an outgoing MAC address and an outgoing label. Figure 4 shows a schematic diagram of the interaction between the host software and the enhanced NIC that contains the label-switching table.

As in basic MPLS, labels are associated with routes or destinations, i.e. at any node, all entries in the label switching table that refer to the same route (the same path to the same destination) will share the same outgoing MAC address (of the next hop) and outgoing label. For example, let an entry in the switching table of B be (A, L_{AB}, C, L_{BC}). This means that any packet received at B from A with a label L_{AB} will use C as the next downstream hop with a label L_{BC}[4].

[3][22] have benchmarked the sources of latency in typical packet handling operations on an MPI architecture using 20 byte packets to about 100μs for NIC-host-NIC (excluding route-lookups and related overheads). Even though these values may be OS and driver specific, we use similar values to the ones described.

[4]The MAC address itself cannot be used as a label, since packets that are received at B need to be further distinguished based on their individual destination. Thus, two identifiers are needed, one for the next hop node and the other for the eventual destination

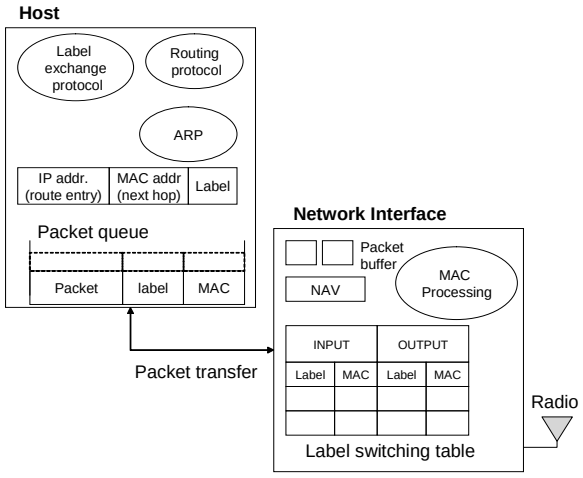

Fig. 4. Host and NIC components for packet forwarding using labels

The combination of the outgoing label L_{BC} and the MAC address of the next hop node C, essentially defines a specific route to a destination, say D. If B has another neighbor, say Z, which uses B to reach D as well, then there will a corresponding entry in the label-switching table (Z, L_{ZB}, C, L_{BC}). The number of distinct outgoing labels is equal to the number of destinations in the network. It should be noted that each label is unique only to a single hop, and the same label may be re-used by different nodes of the network. We shall see that the label information is not needed in the DATA packets, but is carried only in control packets such as the RTS and CTS frames. This is possible because the MAC protocol reserves a time duration (via control packets) during which a forwarding node can expect to receive a DATA packet.

A. Alternative choices for "labels" and their distribution mechanisms

It is important to note that the gain associated with label based switching arises from eliminating the route lookup overhead and replacing it with interface-contained forwarding. The label used for enabling interface based switching can thus be just a simple flow identifier or a combination of MAC address and flow identifier. ICF gains will not be affected by the choice of any particular labeling mechanism. Also, there may be different mechanisms for distributing these labels amongst nodes. In [18], the authors propose a Wireless Ad-hoc Label Switching (WALS) architecture that extends the 802.11 data frame header to carry label and flow information for multi-path setup and forwarding, thereby eliminating separate signaling. In [19], a label routing protocol (LRP) that is used to setup, configure and maintain paths between communicating end-points. Also, 802.11 Wireless Distribution System (WDS) uses nested MAC headers for switching a packet between the source and destination AP's in a network consisting of multiple AP's connected by a wired/wireless backbone. Space for these addresses in the MAC header could potentially be used to carry labels for our proposed scheme. A separate scheme for assigning labels to routes would still be needed since address discovery techniques used in WDS (e.g broadcasting ARP

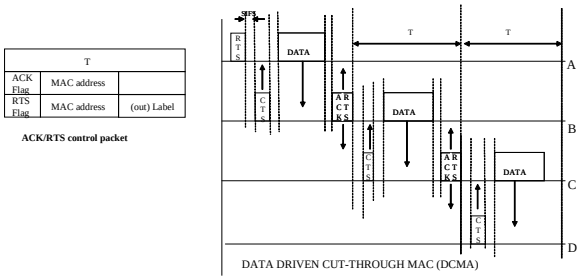

Fig. 5. DCMA Timing Diagram

requests, observing MAC headers of in-flight packets) would not scale to large multi-hop networks.

The overhead for storing the label lookup table at a node also is relatively small. The number of entries in the table is proportional to the number of destinations (or nodes) in the topology. Each entry consists of (Incoming MAC, incoming label, Outgoing MAC, outgoing label). With a 48 bit MAC address and 32 bit label, every entry requires 48x2 + 32x2 = 160 bits for storage. Thus, even for 1000 destinations (an extremely large wireless mesh!), the table size would be $\approx 160Kb$, which is relatively small.

IV. DCMA: The Cut-through MAC Protocol

In this section, we describe the MAC layer enhancements to gain full advantage of the label-switching described in the previous section. Without these enhancements, a packet would need to be buffered at the NIC between the two separate channel accesses thereby nullifying the performance benefits (in terms of latency or throughput) achieved by the elimination of the routing lookups.

Our proposed MAC scheme is based on enhancements to the IEEE 802.11 Distributed Coordination Function (DCF) mode of channel access and follows the associated 4-way handshake involving the exchange of RTS/CTS/DATA/ACK packets. We term this scheme as Data-driven Cut-through Multiple Access (DCMA). DCMA attempts to replace the two distinct channel accesses, upstream and downstream, with a combined access. The reservation for the downstream hop (B to C) is attempted only after successfully receiving the DATA packet from the upstream node (A). The advantage is that a downstream reservation is made only after the upstream channel access has been granted and the packet reception from the upstream node is successful. Accordingly, as shown in Fig 5, DCMA combines the ACK (to the upstream node) with the RTS (to the downstream node) in a single ACK/RTS packet that is sent to the MAC broadcast address.

The payload of the ACK/RTS packet now contains the MAC address of the upstream node, A, and the MAC address of the downstream node, C. It also includes a label intended for use by the downstream node to figure its next hop. Since the downstream node (and all other neighboring nodes of the forwarding node) is assured to be silent until the completion of the ACK, piggybacking the RTS packet provides the forwarding node with preferential channel access for the downstream transmission. Before sending the ACK/RTS, the forwarding node (B) performs channel sensing to check whether the

TABLE I
SUMMARY OF 802.11B/G AND DCMA PARAMETERS

Event	Time(in μs) (with b/ **with g**
SIFS	10 (**10**)
aSlotTime	20 (**9**)
$T_{rtlookup}$ (Route lookup delay at each forwarder)	1000 (**1000**)
DIFS	50 (**28**)
T_{PHY} = (PLCP header + long preamble)	192 (**20**)
T_{RTS} = (T_{PHY} + 20 bytes at 2Mbps) (802.11b) or 6 Mbps (802.11g)	272 (**46.6**)
T_{CTS} = (T_{PHY} + 14 bytes at 2Mbps) (802.11b) or 6 Mbps (802.11g)	248 (**38.6**)
T_{ACK} = (T_{PHY} + 14 bytes at X Mbps)	192 + 192/X (**20 + 192/X**)
T_{DATA} = T_{PHY} + 8*(L+36 byte MAC hdr)/X	192+8*(L+36)/X (**20+8*(L+36)/X**)
$T_{backoff}$ = 0.5*CW_{min}*aSlotTime	310 (**67.5**)
T_{RTS_DCMA} = (T_{PHY} + 20 bytes RTS + 4 byte label) at 2Mbps)	288 (**52**)
T_{ACK_RTS} = (T_{PHY} + 14 byte ACK + 1 byte flag + 4 byte label + 6 byte MAC addr at 2Mbps)	292 (**53.33**)

medium in its vicinity is idle. This reduces the likelihood of backoffs that might be generated at node B when its cut-through request (RTS/ACK sent to C) fails due to hidden node effects (e.g., when a currently transmitting downstream node D prevents node C from responding with a CTS) associated with the discrepancy between channel sensing and transmission ranges. If cut-through does fail; the forwarding node simply queues the packet in the NIC queue and resumes normal 802.11 channel access. DCMA requires no modification of the 802.11 NAV-a node simply stays quiet as long as it is aware of activity involving one or more of its neighbors. Any node that overhears an ACK/RTS not addressed to it merely increments the NAV by the specified time interval; this NAV increment is also performed by the target of the ACK (the upstream node).

In DCMA, the label is carried in the RTS/ACK (or RTS). In principle, the DATA field could also have carried this label, since the label lookup (to find the downstream node) is not strictly necessary until after the DATA is received. However, by carrying the label information in the RTS, we provide the forwarding node additional time to complete the lookup (in parallel with the DATA transfer from the upstream node). Thus, DCMA provides an end-to-end "reservation" for the flow, assuming that all cut-through attempts are successful. IEEE 802.11e [13] standard specifies a frame bursting mode in which, after gaining access to the channel, the sender does not wait for the required DIFS interval between two frames. Instead, it waits only for SIFS and then transmits the second data frame. DCMA can be thought of as an "end-to-end" extension to this bursting mode.

A. Impact on the Latency using cut-through

As stated earlier, the conventional packet forwarding process results in two types of latencies, one associated with the multiple NIC-to-host packet transfers, and the other with the separate independent channel access attempts. While the overhead associated with the NIC-to-host packet transfers

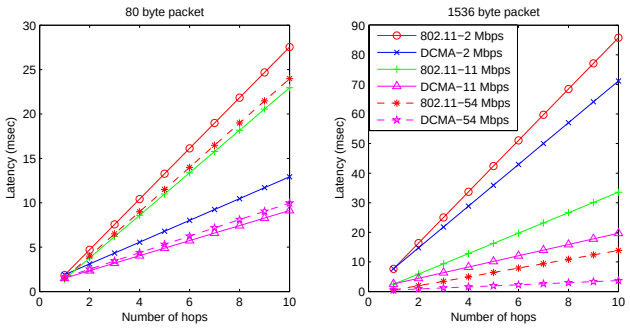

Fig. 6. Latency for different rates and packet sizes (802.11 vs DCMA)

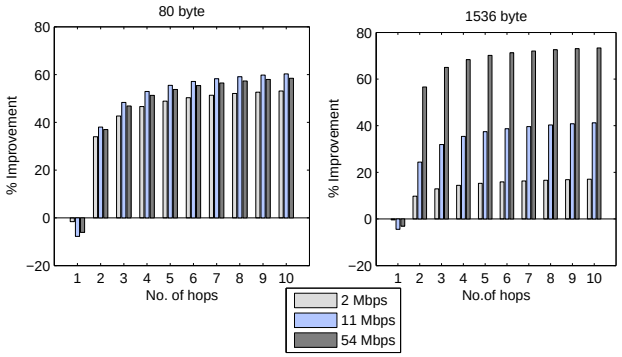

Fig. 7. %improvement in latency at different PHY rates (802.11 vs DCMA)

and route lookups will be hardware and operating system dependent, as explained in section 2.1, we use an upper bound of 1 millisecond at each hop to quantify the total latency in all the required operations. Consider a single data path consisting of N links, defined over the hosts H_1 to H_{N+1}. Let us consider the 802.11b standard and assume that each of the N links can sustain a raw "data" transfer rate of X Mbps (where X is one of 2 Mbps, 11 Mbps and 54 Mbps). To ensure that all stations are able to correctly update their NAV by listening to the signaling packets, the RTS/CTS packets are always sent out at the base rate of 2 Mbps for 802.11b and 6 Mbps for 802.11g while the ACKs are sent out at the data rate. By considering the additional overhead imposed by the PHY layer), we can see that for a MAC layer DATA payload of L bytes, each individual packet transfer consumes a total delay (we ignore propagation delays in our analysis) as shown in Table 1. For 802.11, we assume that each of the transfers over an N-hop path is independent. Hence, the total latency to send L bytes of DATA payload at X Mbps over N hops, is given as follows.

$$\begin{aligned} T_{80211}(N,L,X) &= N*(T_{backoff}+DIFS+T_{RTS} \\ &\quad +T_{CTS}+T_{DATA}+T_{ACK}+3*SIFS) \\ &\quad +(N-1)*T_{rtlookup} \\ &= N*\left[1294+\frac{480+8*L}{X}\right]+(N-1)*T_{rtlookup} \end{aligned}$$

For DCMA, RTS packets have an additional label field (4 bytes) intended for the downstream neighbor. The ACK/RTS

packet is the same as 802.11 ACK with the following additional fields: upstream node MAC address (6 bytes), label for the downstream neighbor (4 bytes), and a flag to indicate ACK/RTS (1 byte).

Now, consider the case of pipelined transfers from H_1 to H_{N+1} using the DCMA protocol. In this case, the channel access delay is incurred only in the first host (original sender). Moreover, now since ACK and RTS share the same frame (on all intermediate hops), the total latency to send L bytes of DATA payload at X Mbps over N hops is given as follows.

$$\begin{aligned} T_{DCMA}(N,L,X) &= T_{backoff} + DIFS + T_{RTS\_DCMA} \\ &+N*(3*SIFS + T_{CTS} + T_{DATA} + T_{ACK\_RTS}) \\ &= 658 + N*\left[762 + \frac{8*(L+36)}{X}\right] \end{aligned}$$

We plot the latency for a 7-hop chain for DCMA and 802.11 with different data rates (2 and 11 Mbps) and different packet sizes (80 bytes and 1536 bytes) in Fig 6. Additionally, latencies using 54 Mbps data rate provided by 802.11g/a cards and the appropriate interframe spaces are calculated using similar analysis as shown above and are also shown in the figure. We also plot the % improvement in latency of DCMA vs 802.11 in Figure 7. The percentage improvement is calculated as $100*(802.11latency - DCMAlatency)/802.11latency$. At higher data rates and packet sizes, the contribution of packet transmission time to the total latency is much smaller compared to the channel access latency. Hence, by reducing the channel access latencies, DCMA will provide a significant improvement in end-to-end latency, especially as *wireless technology evolves to support higher and higher data rates*. As seen in the Fig 7, the improvement in latency for 1536 byte payload transmitted at 54 Mbps is $\approx 73\%$ as compared to $\approx 17\%$ at 2 Mbps.

V. Performance Results

To study the performance of our pipelined forwarding mechanism, we implemented the DCMA protocol as part of the ns-2 simulator [23] with the CMU wireless extensions [24]. We focus on three metrics: a) the throughput improvement achieved by the cut-through protocols b) the potential reduction in end-to-end latency due to the expedited MAC forwarding and c) Percentage of cut-through out of the total packets received. To study the throughput and latency behavior of flows, we ran UDP flows with varying packet arrival rates. The buffer size at each node was 50 packets. The routing tables were pre-configured with the shortest path routes to their respective destinations. Each node keeps track of the number of packets sent out and the number of cut-through acknowledgements received. The cut-through percentage is calculated as the ratio of sum total of the cut through ACKs (ACKs for RTS/ACK-driven transmissions) received to the total number of packet transmissions (each packet transmission on a link is considered separately). The parameters of the ns-2 simulator are summarized in the Table II.

TABLE II
Common Parameters for all simulation results

Topology	7 node chain and 100 node grid
Traffic model	CBR (UDP)
Data Rate	11 Mbps
Transmission/Interference Range	250m/550m
RTS Threshold	0 (always on)
Min. distance between adjacent nodes	248m
Host processing delays at each relay (for 802.11)	1 millisecond

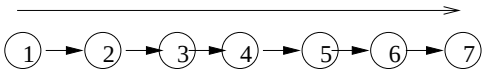

Fig. 8. Simple chain topology with 6 hops and 7 nodes

A. Investigations on the Chain Topology

We first present results of performance studies on a 7-hop chain shown. We vary the offered load and configuration of the flows to understand the various interactions between the cut-through and conventional packets.

1) Single Flow Topology: In this scenario, the traffic consists of a single UDP flow between the two end nodes of the chain. Even though this scenario has only a single flow as shown in Fig 8, it helps to understand the benefits of DCMA over 802.11 under different offered loads and acts as a baseline for further experiments. The offered load was increased from 125 kbps to 1.75 Mbps in steps of 125 kbps using two different packet sizes: 256 bytes and 1536 bytes. Figure 9 shows the throughput and delay results for DCMA and 802.11 for two different packet sizes. We see that for both packet size, the DCMA throughput is higher than 802.11. As expected, DCMA offers almost a 50% reduction in end-to-end delay, especially at higher offered loads.

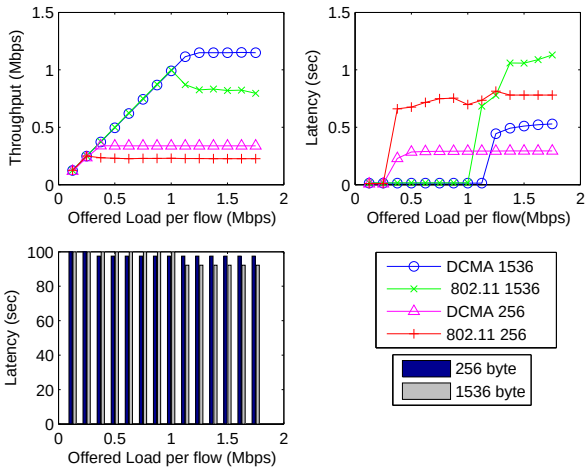

Fig. 9. a. Throughput b. Latency and c. % cut-throughs for a simple chain topology (CBR traffic)

Moreover, the lower throughput for 256 byte packets is due to the proportionally larger overhead of the MAC/PHY-layer headers. One of the most important advantages with DCMA is that the pipelined access mechanism essentially reduces

the channel contention effects among consecutive intra-flow packets - by allowing most packets to cut-through faster to downstream nodes, it lowers the incidence of contention-induced backoffs at upstream (hidden) nodes for subsequent packets. This intra-flow contention becomes especially more pronounced for larger packet sizes with 802.11, where the throughput actually declines from $\approx 1Mbps$ to $\approx 750Kbps$- since each packet transmission now takes longer, each transmitting node now "holds" the channel for a longer duration and may thus cause repeated multiplicative backoff for upstream hidden nodes (an observation also reported in [6]). At relatively low traffic rates, the packets arrive sufficiently spaced apart to avoid this problem of intra-flow contention. We expect the latency difference between 802.11b and DCMA to be much higher for higher data rates (54 Mbps) as shown in Fig 7. Fig. 9c shows the percentage of cut-through packets to the total number of packets delivered from the source to the destination. For lower offered loads, there are 100% successful cut-throughs at all intermediate nodes. However, as the offered load increases beyond 1 Mbps, the packet injection rate is much higher than the cut-through delivery time, resulting in queue build up at the intermediate nodes and reduced cut-throughs.

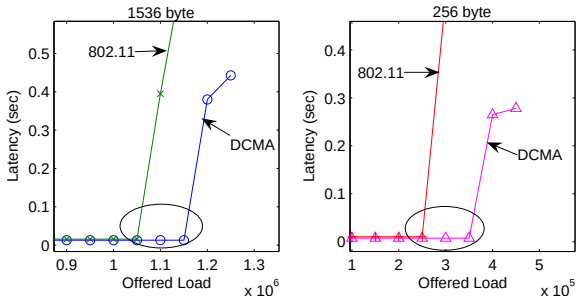

Fig. 10. Zoomed view of flow latencies for a. 256 byte and b. 1536 byte payload

Fig 10 provides a zoomed view of the average delays for each packet size. Note that many representative interactive or multimedia applications require the end-to-end latency not to exceed 100 to 200 ms. Accordingly, we define the *operating range* of a protocol to be that where the end-to-end latency does not exceed 200 msec. As seen in the figure, DCMA extends the operating range from 1.05 Mbps to 1.15 Mbps for 1536 byte packets and from 250 Kbps to 350 Kbps for 256 byte packets while maintaining reasonably low latency.

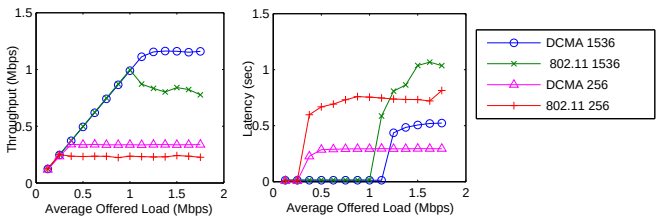

Fig. 11. a. Throughput b. Latency of each flow and c. % cut-throughs for a simple chain topology (Poisson traffic)

We also tested the same scenario with Poisson traffic where the traffic was varied from an average offered load of 125 Kbps to 1.75 Mbps in steps of 125 Kbps. The results for throughput and delay (as shown in Fig 11) for two different packet sizes show similar performance as with the constant bit rate traffic.

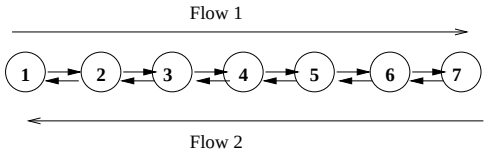

Fig. 12. Chain topology with two reverse flows

2) Simple chain with two flows in reverse direction: We next considered two flows in reverse direction (from node 1 to 7 and from node 7 to 1 respectively, as shown in Fig. 12). This scenario also provides insights into the behavior of TCP traffic, which has the data flowing from the source to the destination and ACKs flowing in the reverse direction. In this case, the offered load was increased from 125 kbps to 1 Mbps per flow. Only the results for the 256 byte packets are presented for reasons of space).

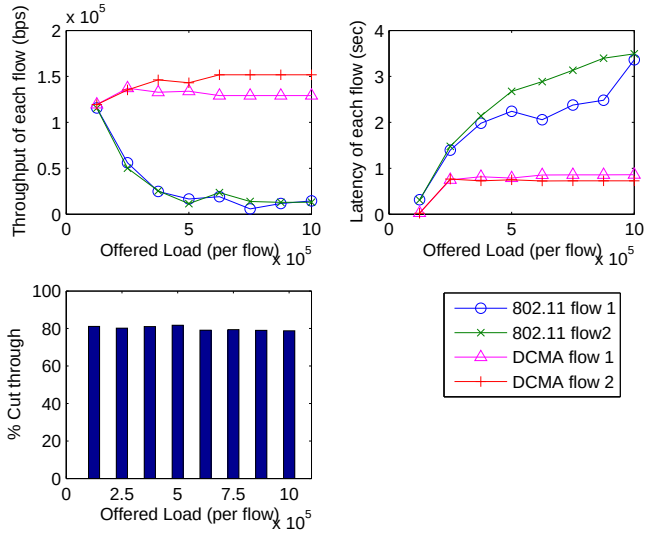

Fig. 13. a. Throughput b. Flow Latencies and c. % cut-throughs for two flows in reverse direction

Fig 13 plots the throughput, end-to-end latency and percentage cut-through rates for DCMA and 802.11 for this scenario. Once again, we see that DCMA is able to obtain significantly higher (almost double the throughput of 802.11) in this case. Note that the two flows have similar throughput and delay values indicating that each flow gets a fair allocation of the channel for both 802.11 and DCMA. It is also important to note that the sources of traffic (nodes 1 and 7) do not participate in the relaying process.

3) Simple chain with two flows in the same direction: Next, we look at a case where the two flows are in the same direction (1 to 7 and 4 to 7 as shown in Fig 14). This scenario deals with the case when the intermediate relay nodes may also have traffic of their own to send. We study the per flow behavior with 802.11 and DCMA using the same traffic profile as described in V.A.2.

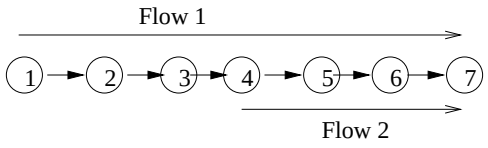

Fig. 14. Simple chain (flows in same direction)

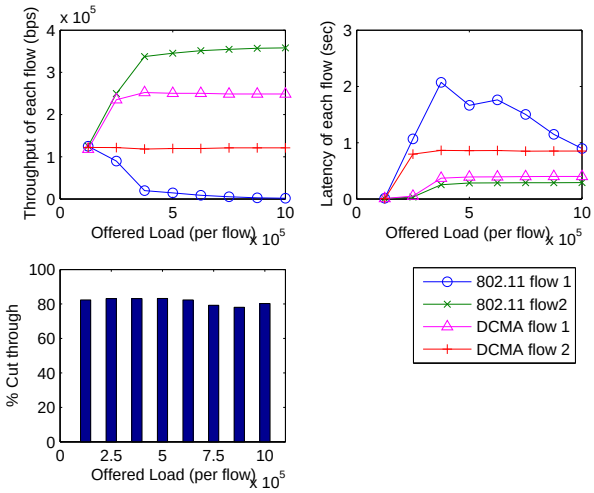

Fig. 15. a. Throughput b. Flow latencies and c. % cut-throughs for two flows in same direction

The results in Fig 15 show that for both 802.11 and DCMA, one of the flows is starved, resulting in disproportionate throughput and average delays. Note that even in this case, DCMA still performs slightly better since by using efficient cut-throughs, the packet is delivered faster, resulting in an earlier channel access for the starved flow when the medium is idle. In Section V.C, we propose a simple heuristic algorithm to address this starvation problem.

B. Investigations with the Grid Topology

After studying the chain topologies described earlier, we tested DCMA on a general 10×10 grid topology with randomly selected sources and destinations. The parameters used in the simulations are summarized in Table III.

TABLE III

PARAMETERS FOR THE GRID TOPOLOGY

Number of flows	5, 10, 15, 20, 25 30
Packet sizes (bytes)	256
Dimensions	2500m×2500m
Distance between adjacent nodes	248 m

Thus, the grid topology was a combination of the scenarios described earlier (multiple flows - in the same direction, reverse directions as well as interfering flows). We showed earlier in the simple chain single flow topology (Fig 9) that for 256 byte packets, the performance of DCMA and 802.11 diverges at an offered load of around 200-400 Kbps per flow. This can be attributed to the fact that for low offered loads, queues are usually small or empty and the major component of the end-to-end delay is thus the packet exchange times

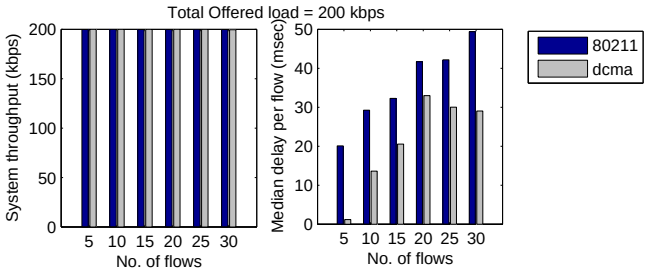

Fig. 16. Lightly loaded case: System throughput and median delay per flow

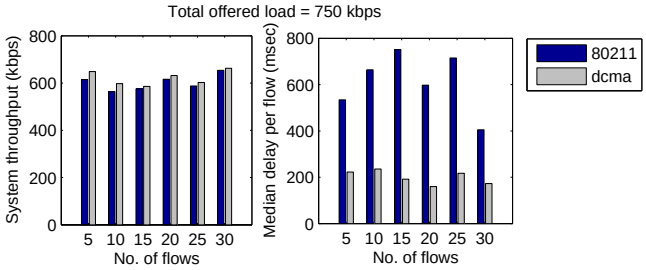

Fig. 17. Medium loaded case: System throughput and median delay per flow

between nodes (which is almost the same for both 802.11 and DCMA). At offered loads of about 200-400 Kbps, there are enough packets in the pipe to keep the channel busy. Here, the performance of 802.11 and DCMA diverges as the channel access delays become comparable to the packet transmission times. DCMA outperforms 802.11 as it saves one channel access at every intermediate node as compared to 802.11 that needs two distinct channel accesses. Note that the operating load per flow in the grid scenario might be lower than in the single flow chain, due to contention among packets belonging to different flows that have intersecting paths. In our simulations, we only look at the useful operating range where the end-to-end latencies are within 1 second. We consider two different cases, as described below, both for lighly loaded and medium loaded traffic. For all the cases, we ran the simulation for 20 topologies and present the system throughput and the median delays per flow.

1) Constant total offered load, increasing number of flows: In this case, the total offered load to the grid was kept constant at 200 kbps and 750 kbps representing the lightly loaded and medium loaded conditions. Across the simulation runs, we varied the number of flows (from 5 CBR flows to 30 flows). As shown in Fig 16, for the lightly loaded case, both DCMA and 802.11 have similar performance. As the number of flows increases, there is a higher probability that flows interfere with one another due to common relays along their paths. At lower loads, DCMA still outperforms 802.11 since cut-throughs allow a packet to be delivered faster, resulting in an earlier medium access for competing flows. Also, it can be seen that for the medium loaded case, as shown in Fig 17, 802.11 latencies are almost three times the corresponding DCMA latencies. Thus, DCMA outperforms 802.11 for both the light and medium loaded cases.

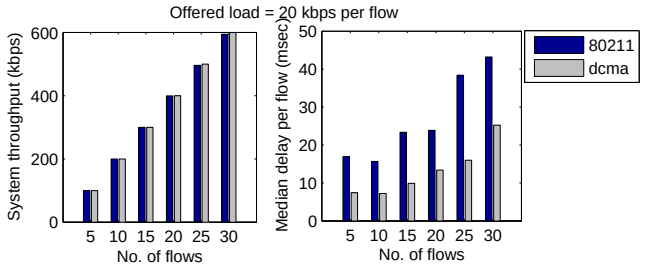

Fig. 18. Lightly loaded case: System throughput and median delay per flow

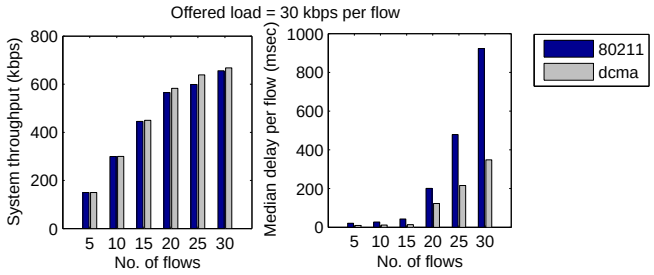

Fig. 19. Medium loaded case: System throughput and median delay per flow

2) Constant offered load per flow, increasing number of flows: In this case, the individual offered load per flow was kept constant (20 kbps and 30 kbps) and the number of flows was increased from 5 to 30. Thus, the total offered load was increased from 100 kbps to 600 kbps and 150 to 900 kbps for the two cases respectively. As seen in Fig 18, both DCMA and 802.11 have low latencies at lighter loads. As the offered load per flow increases, the number of successful cut-throughs reduces and the delays increases. However, as seen in Fig 19, DCMA maintains the latencies to within 250ms as compared to the high latencies of 450-850ms for the case of 802.11 due to repeated collisions and backoff.

C. Heuristic approach to address flow starvation and related unfairness

In all our simulations, we have given preferential access to a cut-through when forwarding a packet to the next hop. One potential problem of the preferential access provided to a cut-through flow is that it may cause starvation for other flows, since the NIC may respond to ACK/RTS requests while it has packets of its own to send in its buffer. As a simple fix to this problem, we modified the interface behavior to monitor if the node had any packets pending in its MAC buffer that belonged to the flow originating at this node itself. If so, the NIC would decline the cut-through attempt by any other flow by not sending a CTS in response to an RTS/ACK request from a neighbor. Although our simulations are based on UDP sessions, this fairness policy is especially useful for TCP flows, since it protects against out-of-order delivery of packets. Without the fairness measure, a packet from a flow could get queued at an intermediate node, while a subsequent packet could cut-through to the destination node. Fig 20 shows the per-flow throughput and delay before and after applying the fairness policy for two flows on a simple chain topology

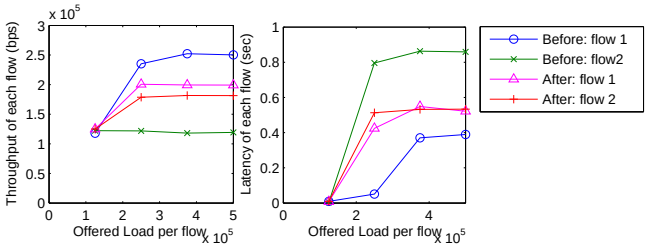

Fig. 20. Per flow throughputs and delays (before and after fairness policy)

as described in Section V.A.3. We can see that with fair-DCMA, individual flow throughputs and delays are much closer than with DCMA without fairness. Note that other fairness policies may also be applied; however, we defer the detailed investigation of various fairness schemes and their impact on system performance to a future paper.

VI. Conclusion

In this paper, we presented an efficient Interface Contained Forwarding architecture for a "wireless router", i.e. a forwarding node with a single wireless NIC in a multi-hop wireless network that allows the process of packet forwarding to be confined entirely within the network interface card. The ICF architecture uses an enhancement to the base 802.11 DCF behavior, using a combined RTS/ACK transmission to reduce the number of channel accesses at the forwarding nodes. Simulation results and analyses demontrate the significance of DCMA's atomic MAC-layer forwarding scheme–even if future hardware improvements almost eliminated the NIC-host-NIC transfer latency, the absence of a pipelined forwarding mechanism in base 802.11 would still result in substantially high end-to-end latency in multi-hop wireless paths, even with ever-increasing channel data rates.

Extensive simulation studies show that DCMA performs better than 802.11 DCF in almost all scenarios. In particular, on simulations over a grid-like wireless mesh, DCMA was able to extend the useful operating range (traffic loads such that end-to-end delays for flows stay bounded below ≈ 200 ms) by more than 30%, compared to 802.11. In general, packet cut-through is especially useful at relatively low to moderate network loads (which is typically the case in well-provisioned wireless networks). Thus, a combination of DCMA and call admission control (so that the network load stays within specified bounds) could prove to be especially useful for relatively low-bandwidth, delay sensitive applications such as VoIP-over-wireless. We also identified an important unfairness issue that might arise in DCMA, where cut-through transmissions of one flow might preemptively starve the transmission of packets belonging to other flows. As a simple solution to this issue, we have proposed a fairness scheme that uses queue occupancy information at the MAC layer to allow competing flows to get equitable channel access. Simulation results show that this simple modification of base DCMA significantly improves the fairness across flows.

There are many interesting ideas for future research on the idea of the wireless router. Our studies indicate that *intra-flow* packet contention can undermine the benefit of packet

pipelining; this suggests that multi-path interleaved routing (where successive packets are sent on link-disjoint paths) could be especially useful with DCMA. Additionally, the pipelining scheme can be even more effective if multiple packets could be pipelined in bursts; this suggests research into techniques for "cumulative" packet cut-through schemes.

REFERENCES

[1] "Champaign Urbana Community Wireless Network." http://www.cuwireless.net.

[2] "Locustworld Mesh Project." http://www.locustworld.org.

[3] "IEEE 802.11 Task Group S Mesh Networking Standard." http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm.

[4] R. Karrer, A. Sabharwal, and E. Knightly, "Enabling large-scale wireless broadband: The case for TAPs," in *HotNets-2*, Nov 2003.

[5] S. Xu and T. Saadawi, "Does IEEE 802.11 MAC protocol work well in multihop wireless ad-hoc networks?," in *IEEE Communications Magazine*, June 2001.

[6] J. Li, C. Blake, D. DeCouto, H. Lee, and R. Morris, "Capacity of adhoc wireless networks," in *ACM International Conference on Mobile Computing and Networking*, Aug 2001.

[7] M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multihop networks," in *IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[8] J. So and N. Vaidya, "Multi-channel MAC for ad-hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing, Mobihoc*, May 2004.

[9] "MPLS Label Switching Architecture." IETF RFC 3031.

[10] C. Perkins and E. Belding-Royer, "Ad hoc On-Demand Distance Vector Routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA*, pp. 90–100, February 1999.

[11] "LDP specification." IETF RFC 3036.

[12] B. Jabbari, P. Papneja, and E. Dinan, "Label switched packet transfer for wireless cellular networks," in *IEEEWireless Communications and Networking Conference)*, August 2000.

[13] "IEEE 802.11e Wireless Medium Access control (MAC) and Physical layer specifications: Medium Access Control (MAC) Quality of Service(QoS) Enhancements," August 2004.

[14] X. Yang and N. Vaidya, "A Wireless MAC Protocol using Implicit Pipelining," in *IEEE Transactions on Mobile Computing*, vol. 5, pp. 258–273, March 2006.

[15] X. Yang, N. Vaidya, and P. Ravichandran, "Split-channel pipelined packet scheduling for wireless networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 3, pp. 240–257, 2006.

[16] P. Kyasunur, J. Padhye, and P. Bahl, "On the efficacy of separating control and data into different frequency bands," in *IEEE Broadnets*, October 2005.

[17] A. Acharya, A. Misra, and S. Bansal, "A label-switching packet forwarding architecture for multi-hop wireless lans," in *5th ACM International Workshop on Wireless Mobile Multimedia (WoWMom), in conjunction with MobiCom 2002*, Sept 2002.

[18] H. Liu and D. Raychaudhuri, "Label switched multi-path forwarding in wireless ad-hoc networks," in *Third IEEE International Conference on Pervasive Computing and Communications, PERCOM*, pp. 248–252, March 2005.

[19] Y. Wang and J. Wu, "Label routing protocol: a new cross-layer protocol for multi-hop ad hoc wireles networks," in *Proceedings of the International Conference on Mobile Adhoc and Sensor Systems Conference, MASS*, pp. 139–145, November 2005.

[20] D. Raguin, M. Kubisch, H. Karl, and A. Wolisz, "Queue-driven cut-through medium access in wireless ad hoc networks," in *IEEEWireless Communications and Networking Conference)*, March 2004.

[21] "The journey of a packet through the linux 2.4 stack network." http://gnumonks.org/ftp/pub/doc/packet-journey-2.4.html.

[22] W. Zhu, A. Maccabe, and R. Riesen, "Identifying the sources of latency in a splintered protocol," tech. rep., Department of Computer Science, University of New Mexico, Dec 2003.

[23] K. Fall, "ns notes and documentation," *The VINT Project*, 2000.

[24] "The CMU Monarch Project's Wireless and Mobility Extensions to NS," 1998.

Arup Acharya Dr Arup Acharya is a Research Staff Member in the Internet Infrastructure and Computing Utilities group at IBM T.J. Watson Research Center and leads the Advanced Networking micropractice in On-Demand Innovation Services. His current work includes SIP-based services such as VoIP, Instant Messaging and Presence, and includes customer consulting engagements and providing subject matter expertise in corporate strategy teams. Presently, he is leading a IBM Research project on scalability and performance of SIP servers for large workloads. In addition, he also works on different topics in mobile/wireless networking such as mesh networks. He has published extensively in conferences/journals and has been awarded seven patents. Before joining IBM in 2000, he was with NEC C&C Research Laboratories, Princeton. He received a B.Tech degree in Computer Science from the Indian Institute of Technology, Kharagpur and a PhD in Computer Science from Rutgers University in 1995. Further information is available at http://www.research.ibm.com/people/a/arup/

Sachin Ganu Sachin Ganu is currently a Ph.D. candidate at WINLAB, Rutgers University working under the guidance of Dr. D. Raychaudhuri. His research interests include cross-layer architectures for wireless mesh networks focusing on MAC-routing interactions and location estimation in wireless networks. Currently, he is also involved in the design and development of a large scale indoor wireless testbed, ORBIT, to facilitate wireless experimentation. Prior to joining WINLAB, Sachin worked as a Systems Engineer with Lockheed Martin Global Telecommunications, Clarksburg, MD from Feb 2000 until Dec 2001 on radio resource management tools for ATM-based geosynchronous satellite systems. He worked as a Research Scientist at Avaya Research Labs, Basking Ridge, NJ during the summer of 2002 and as a Network Software Engineer with the Communications Technology Group at Intel Corporation, Hillsboro, OR during the summer of 2005. Sachin received the B.E degree in Electronics from Mumbai University, Mumbai, India in 1998, and his M.S degree in Electrical Engineering from Virginia Tech, Blacksburg, VA, in 1999.

Archan Misra Archan Misra is a Research Staff Member at the IBM TJ Watson Research Center, Hawthorne, NY, where he works on infrastructural components for pervasive networking and middleware for presence-based, context-aware computing. His current research projects include middleware for SIP-based "rich presence" in converged networks, high-performance wireless mesh networks and sensor data management techniques for healthcare and remote monitoring scenarios. Archan received his Ph.D. in Electrical and Computer Engineering from the University of Maryland at College Park in May, 2000, and his B.Tech in Electronics and Communication Engineering from IIT Kharagpur, India in July 1993. Prior to joining IBM in 2001, Archan worked at Telcordia Technologies on architectures for QoS in the Internet and adaptive IP-based mobility management algorithms. He has published extensively in the areas of wireless networking and pervasive computing and is a co-author on papers that received the Best Paper awards in ACM WOWMOM 2002 and IEEE MILCOM 2001. He is presently on the editorial board of the IEEE Wireless Communications Magazine, and currently chairs the IEEE Computer Society's Technical Committee on Computer Communications (TCCC).