RC23210 (W0405-062) May 12, 2004
Computer Science

IBM Research Report

DCMA : A Label Switching MAC for Efficient Packet Forwarding in Multi-Hop Wireless Networks

Arup Acharya, Sachin Ganu*, Archan Misra
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*WINLAB
Rutgers University
Piscataway, NJ 08854

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

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

Arup Acharya — *IBM T.J.Watson Research Center, Hawthorne, NY*

Sachin Ganu — *WINLAB, Rutgers University, Piscataway, NJ 08854*

Archan Misra — *IBM T.J.Watson Research Center, Hawthorne, NY*

Abstract

This paper addresses the problem of efficient packet forwarding in a multi-hop, wireless "mesh" network. We first explain how the packet forwarding operation in the wireless broadcast medium differs from that of a router in wired network, which typically switches packets across multiple network interfaces. In a wireless mesh network on the other hand, any node with a single wireless network interface card can operate as a router or a forwarding node, since it can receive a packet from a neighboring node, do a route lookup based on the packet's destination IP address and then transmit the packet to another neighboring node using the same wireless interface. This paper introduces 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 DCMA, which preserves all the basic distributed medium access functions of 802.11. DCMA's primary innovation lies in the use of MPLS-like labels in the control packets, and in the use of a combined ACK/RTS packet to combine upstream packet reception acknowledgment with downstream channel access. Using extensive simulations with two different topologies: chain and grid, we compare the performance of DCMA with 802.11 DCF MAC with respect to throughput and latency.

1. INTRODUCTION

The raw channel speeds for the IEEE 802.11 [1][2] family of standards are increasingly rapidly: while the 802.11a standard already operates at up to 54 Mbps, enhanced versions operating at speeds up to 108 Mbps are under active investigation. A variety of popular initiatives are exploring the use of 802.11, or minor

variants of it, for constructing extended multi-hop wireless *mesh* networks [3]. If the raw channel capacity of the basic 802.11 channel could be effectively utilized, *multi-hop, wireless* networks, where the wired backbone is reachable only via multiple wireless hops, can indeed become a compelling low-cost broadband access alternative, especially in relative dense urban areas [4]. Potential examples of such wireless meshes include *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 communities.

The throughput achieved by current implementations of multi-hop 802.11 networks is, however, significantly lower than the underlying channel capacity. Some of the reasons for this significant drop in overall 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 has been suggested in literature, and continues to be the subject of active investigation.

In this paper, we however 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 distinct from the corresponding operations performed in a wired network:

> *In a wired network, a forwarding node typically[1] 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, a node N, with a single wireless interface, may act as a forwarding node by transmitting a packet to a node other than from which received the*

[1] Overlay networks could be created out of tunnels using single network interface cards.

packet. In effect, N acts as an intermediary for two nodes that are each within the communication range of N but not directly within the range of each other.

Due to this unique facet of spatial diversity, 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.

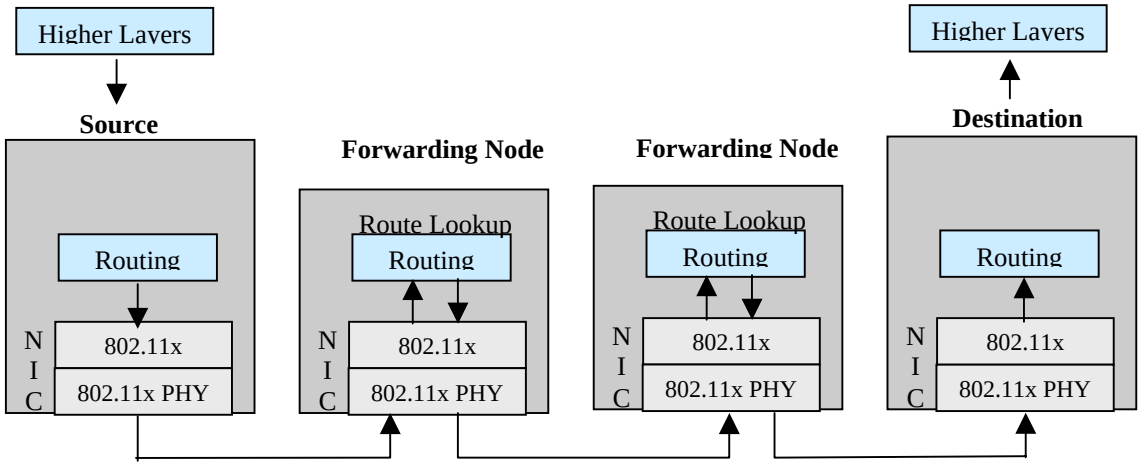


Figure 1 Typical packet forwarding in a multi-hop wireless network

This mechanism introduces two forms of *latency* in the multi-hop wireless forwarding process that is independent of all the other 802.11-related drawbacks enumerated earlier:

1. The forwarding node is thus involved in two separate channel access attempts during the forwarding process: once to receive the packet and again to "forward" it. For each channel attempt, the contention

[2] In high-end routers/switches, the packet is transferred from one interface to another via a dedicated switching fabric, while in software based routers, the packet is processed by the host CPU (e.g. route lookup) between packet reception on one interface and subsequent transmission on another.

resolution mechanism requires the transmitting node to perform an initial random backoff (to avoid grabbing the channel completely), even if no other node in the transmitter's neighborhood has any packet to send.

2. The forwarding operation involves two separate transfers of data between the memory on the network interface card (NIC) and the host's memory (accessed by the host software), and must thus suffer from the latencies associated with interrupt-handling, packet copying and route lookup.

As we shall demonstrate later, these two artifacts can significantly increase the overall end-to-end latency of packet delivery over a wireless mesh, especially at low system loads, and thereby contribute to a lower overall system throughput. Our primary goal in this paper is thus to define a practical architecture and protocol to avoid the two forms of latency mentioned above. 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) 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 form of efficient "layer-2 forwarding", called Interface-Contained Forwarding (*ICF*), is feasible in practice and can be enabled by using MPLS-like [9] 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 such as LDP [10] to distribute levels to appropriately reflect the traffic routes. This allows packet forwarding to be confined *entirely to the NIC*, which matches the label of an incoming packet with an entry in the 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 necessary to also 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. We shall thus 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, while remaining fully conformant to the basic 802.11 sensing and backoff primitives. Both of these enhancements

work in tandem: to exploit this "cut-through" capability of the MAC layer, the NIC must be capable of determining the identity of the next-hop node (without invoking a lookup of the routing tables resident in the host protocol stack) from the signaling information included in the contention resolution phase. Moreover, the atomic channel access scheme must continue to provide some form of fair channel access to all nodes, and must especially guarantee protection from starvation to any other flow that contends for resources on this pipelined forwarding path.

Our focus in this paper is only on static wireless multi-hop networks; accordingly, the choice of an appropriate routing protocol to determine an appropriate path between two end-points on the wireless mesh is outside the scope of this paper. Clearly, any suitable ad-hoc routing protocol, such as DSR [11] or AODV [12], may be used to set up the appropriate routing tables at each node.

The rest of this paper is organized as follows. Section 2 presents a more detailed overview of the basic 802.11 forwarding process and quantifies the latencies involved. 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 enabled atomic packet forwarding in wireless networks. Section 5 then describes our implementation of DCMA within 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 quantifies how DCMA essentially degrades to base 802.11 as the network becomes heavily loaded. Finally, Section 6 concludes with a discussion and summary of future work and open research issues.

1.1 Related Research

The use of MPLS (or labels) for providing fast and efficient packet forwarding on the wireless channel has not been extensively reported in literature. The use of MPLS to support packet routing and handoff in wireless cellular networks is proposed in [13] 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, can be viewed as similar to the fragment burst mode [1] of 802.11, where a sender splits up a larger packet into multiple back-to-back frames, with each frame reserving the channel for the successive frame. While the fragment burst mode is designed for reducing the impact of channel errors on a single link, DCMA is designed for faster data transfer over *multiple consecutive* links in a multi-hop network.

2. CONVENTIONAL PACKET FORWARDING UNDER 802.11

In this paper, we focus on 802.11's *Distributed Coordination Function (DCF)* contention resolution as it is commonly employed in multi-hop ad-hoc networks, where each node essentially acts a peer to all nodes within its transmission range. Unicast communication in the DCF mode involves a 4-way handshake mechanism (shown in Fig. 2) between a data sender node *A* and the corresponding recipient node *B* to both avoid collisions and verify reliable packet forwarding:

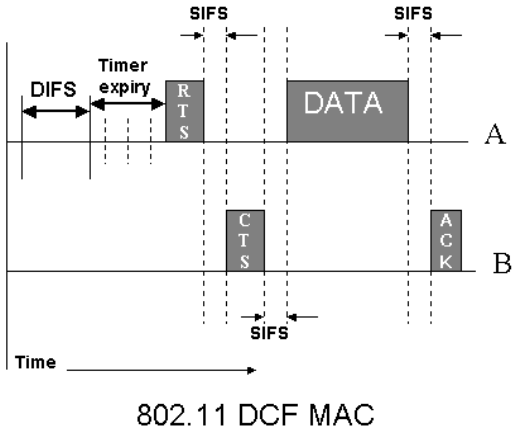


Figure 2 802.11 DCF Channel Access Timers

1. The RTS (request-to-send), sent by node A, specifies a time interval T_{RTS} that includes B's response through a CTS (clear-to-send), followed by data transmission by A and time to send an ACK from A

to B[3]. This is in effect informs anyone within A's neighborhood that the medium is "reserved" for the duration T_{RTS}.

2. The CTS, sent by node B, specifying the time interval T_{CTS} during which A is permitted to send this data--in 802.11, the interval specified in the CTS is equal to the transmission times of the data and the ACK. The CTS informs all neighbors of B that the channel is reserved for the duration T_{CTS}.
3. The data itself, sent by node A, during the slot reserved for it by the CTS--this data transfer phase immediately follows the reception of the CTS. Note that the data transmission interval is typically larger that than control message transmission times (CTS/RTS/ACK). The max data frame that can be sent is 2346 bytes, while the RTS, CTS and ACK control frames are 20, 14 and 14 bytes respectively.
4. The final data ACK, sent by node B, indicating successful reception--this ACK is sent after the end of the transmission of data by A.

For contention resolution, 802.11 uses a timer-based exponential back-off scheme, as follows. Prior to transmitting a packet, a node senses the channel for a period equal to DIFS (Distributed Inter-Frame Space). If the channel is busy, the node selects a random back-off time in the range (0, Congestion Window) (specified in terms of slots). An important observation in this respect is that although the 802.11b standard defines multiple rates for transmission of the data frames, *the slot duration is defined in terms of the base (1 Mbps) rate, and is identical for all nodes, independent of the channel quality*. Also, to ensure that a particular node does not completely monopolize the channel, 802.11 requires each node to perform an *gratuitous backoff* even before the initial DIFS sensing, effectively creating a minimum period of silence between two consecutive transmissions by a single node. The backoff timer is decremented whenever the channel is free; the node makes a fresh attempt at sending an RTS packet upon the expiration of the timer. Upon failure of the RTS packet (no CTS packet is received), the congestion window is doubled and a random timer is chosen from the new window. In addition to the backoff timer, the 802.11 MAC requires every transmission of an RTS packet

[3] The interval also includes short inter-frame spacing (SIFS) periods between the RTS and CTS, CTS and DATA, DATA and ACK. For the rest of the paper, a SIFS period is implied when referring to such transmissions even if not stated explicitly.

to be preceded by a channel sense for DIFS duration, thereby reducing the probability of collision with an ongoing transmission. Each 802.11 is 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.

2.1 Forwarding Operation in 802.11 MAC

We now discuss the overheads associated with a forwarding operation when using the 802.11 MAC in a multi-hop wireless environment. The terms **upstream** node, **forwarding** node and **downstream** node are defined as follows: the upstream node sends a data packet to the forwarding node; which then forwards the packet to the downstream node. Thus, the forwarding node and the downstream node act as the destination for transmissions from the upstream node and the forwarding node respectively. Consider the case shown in Fig. 3, where A is the upstream node, B is the forwarding node and C the downstream node. After the IP lookup function in host A determines that B is the next hop of the DATA packet, 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. Note that node C (and all other neighbors of B) is guaranteed to remain silent during the DATA and ACK portions of this packet transfer, since the CTS from B to A effectively reserves the channel and blocks any concurrent transmission attempt. At B, the packet is transferred to the main memory from the NIC, and the host CPU is notified (e.g. via interrupts) for further processing of the packet by the IP protocol stack running on the host CPU.

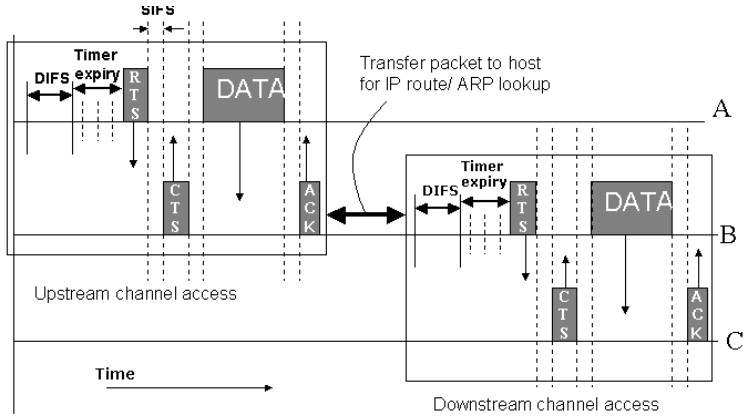


Figure 3 Multi-hop forwarding in 802.11 MAC

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. 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.

3. THE LABEL-BASED ICF ARCHITECTURE

The Interface Contained Forwarding (ICF) architecture uses MPLS-based labels as a technique to avoid the software overheads of NIC-to-host packet copies at each intermediate node on the forwarding path. MPLS usage is primarily by network service providers. MPLS allows service providers to offer a VPN service with QoS (DiffServ), including intranets and extranets. MPLS improves the efficiency of the packet forwarding process by replacing route lookup for a variable length IP destination address, with an exact match of a fixed, predefined number of bits called labels. On inspecting the entire forwarding process between the upstream

node and the downstream node in Figure 3, we observe that considerable performance gains may be expected if the forwarding node's (B's) NIC is able to internally redirect the packet received from the upstream node A back onto the channel towards the downstream node C. To achieve this, B's NIC must be capable of resolving the identity of the downstream node (and its MAC address) directly without resorting to an IP lookup in the host kernel. We defer the exact details of how B obtains label information to determine C's identity to the description of DCMA in Section 4. For now, we explain how such fast-forwarding can be achieved through the use of MPLS-based labels. To support MPLS-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.

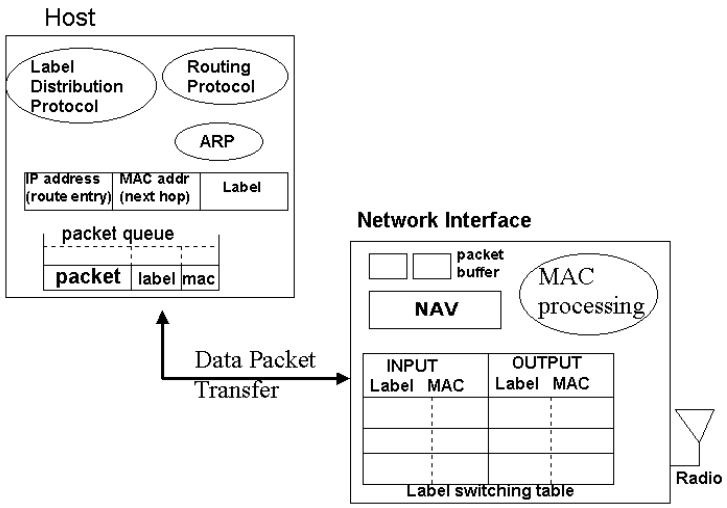


Figure 4 Host and NIC components for packet forwarding using labels

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} >. The interpretation of this entry is 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] 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 Z. If B has another neighbor, say D, which uses B to reach Z as well, then there will a corresponding entry in the label-switching table <D, L_{DB}, 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.

When a packet is forwarded by node B, the incoming label will be replaced by the outgoing label. We shall see that we 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.

It is important to realize that the algorithm for populating of the label-forwarding table is itself outside the purview of the ICF framework. Clearly, the distribution of labels on a specific path requires interaction between the routing layer and MPLS-specific protocols. For example, it is fairly straightforward to piggyback labels with route updates or run a separate label distribution protocol, as is the case for wired networks, e.g. [10]. As a result of the routing and label distribution protocols, the host maintains, in addition to its ARP cache, a queue of packets waiting to be moved to the wireless interface card for transmission onto the wireless channel. Each packet is associated with a (outgoing) label and the MAC address of the next hop node. A packet is placed in the queue when (a) the host generates a packet or (b) when a cut-through terminates at this node, either because the cut-through could not be extended beyond this node because the channel access for the next hop was not successful, or because this node is the final destination of the packet. Prior to inserting each packet in the queue, the host does an IP lookup using the packet's destination IP to determine the packet's next hop node. In addition, the ARP cache is inspected to determine the MAC address of the next hop node, and a route (destination IP) to label mapping table is used to determine the (outgoing) label. Then

[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.

packets are handed over to the NIC one at a time, along with the outgoing label and next-hop's MAC address. In general, the NIC does not need to maintain a packet queue; the packet buffer shown in Fig. 4 is use to hold a packet awaiting channel access, and to buffer a packet while it is in the process of being forwarded. Packets that are successfully forwarded need to be buffered only between reception (from an upstream) node and immediate transmission to the downstream node. If the forwarding fails, i.e. the cut-through did not succeed, or an ACK was not received for DATA transmission, the packet is sent to the host and inserted at the back of the packet queue in the host.

The overhead for storing the label lookup table 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 48×2 + 32×2 = 160 bits for storage. Thus, even for 1000 destinations (an extremely large wireless mesh!), the table size would be ~160 Kb, which is relatively small.

4. DCMA: THE CUT-THROUGH MAC PROTOCOL

We have seen in the last section how the presence of label information in the "data" stream helps the forwarding node's NIC to correctly identify the identity (and the associated label) of the downstream node. Without additional enhancements at the MAC layer, such a packet would however need to be buffered at the NIC between the two separate channel accesses (depicted in Fig 3) until the channel is again acquired for transmission to the downstream node. The resulting latencies (which can be of the order of milliseconds or even seconds if multiple backoffs are involved) can effectively negate any performance benefits (in terms of latency or throughput) achieved by the elimination of the routing lookups. We now explain how our proposed extension to the 802.11 DCF channel access scheme is designed to allow the forwarding node to combine the two separate access channels depicted into a single "seamless" access.

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 is attempted only after successfully receiving the DATA packet from the upstream node. 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, 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, and the MAC address of the downstream node. 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 till the completion of the ACK[5], piggybacking the RTS packet provides the forwarding node with preferential channel access for the downstream transmission. Cut-through in DCMA fails when the downstream node fails to respond to the ACK/(RTS) with a positive CTS; the forwarding node then simply queues the packet in the NIC queue and resumes normal 802.11 channel access. Since the forwarding node (node B in Figure 3) is guaranteed to have a silent neighborhood until the transmission of its ACK, and since DCMA reverts back to 802.11 on failure of the pipelined RTS, DCMA introduces no additional form of channel contention over that of base 802.11. DCMA also requires no modification of the 802.11 NAV—a node simply stays quiet as long as it is aware of (contiguous) 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).

The operation of DCMA can be understood by following the timing diagram provided in Fig. 5. Assume that node A has a packet to send to node D. A[6] sends a RTS to B, which includes a label L_{AB} associated with the route to D. Assuming that its NAV is not busy for the proposed transmission duration, B replies with CTS. B receives the DATA packet, and then sends a RTS/ACK control packet, with the ACK part

[5] This was discussed in section 2.

[6] We assume that the initial IP address to label mapping is done by the sending host, which sends the outgoing label, the MAC address of the next hop and the packet , to its wireless interface card.

addressed to A, and the RTS part addressed to C, along with a label L_{BC}. C's actions would be analogous to B, except that it uses the label L_{CD} in its RTS/ACK message.

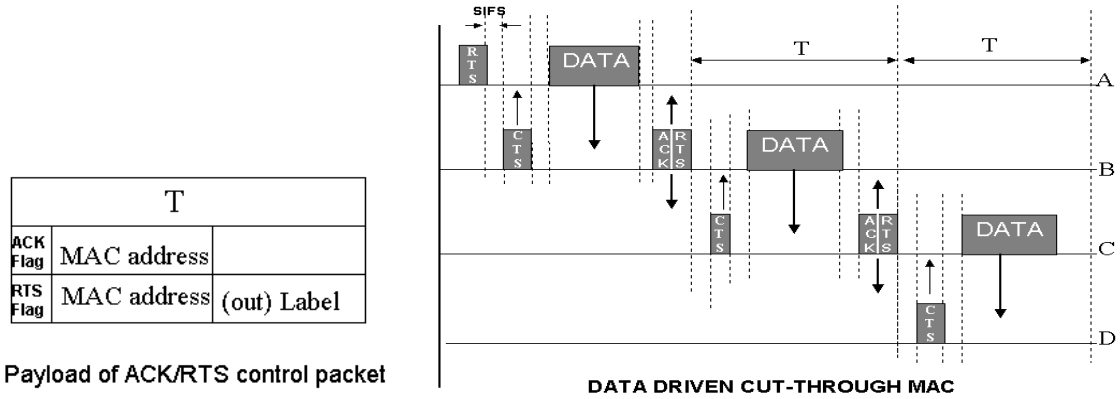


Figure 5 DCMA Timing Diagram

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 providing 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). This should not be a problem, since the DATA duration is at least tens of μsecs (e.g., a 500 byte packet on 2Mbps channel takes 2 msecs).

4.1 Impact of Latency on the Forwarding Process

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 will be hardware and operating system dependent, we can numerically analyze the impact of the independent channel accesses. To study this effect, 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 2Mbps, 5.5 Mbps and 11 Mbps). As explained earlier, each 802.11 transfer involves the 4-way RTS-CTS-DATA-ACK handshake. To ensure that all stations are able to correctly update their NAV by listening to the RTS/CTS/ACK signaling packets, these packets are always sent out at the base rate of 2 Mbps.

By considering the additional overhead imposed by the PHY layer (including the PLCP preamble and the PLCP header which are sent out at 1 Mbps), 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)

SIFS = 10 µs, aSlotTime = 20 µs

*DIFS =SIFS + 2 × aSlotTime = 10 + 2*20 = 50 µs*

T_PHY (PLCP preamble + header=192 bytes @1Mbp) = 192 µs

T_RTS (T_PHY + 20 bytes @ 2 Mbps) = (192 + 80) =272 µs

T_CTS= T_ACK = (T_PHY + 14 bytes @ 2 Mbps) = (192 + 56) =248 µs

T_DATA = (T_PHY + 8× (L+28 bytes MAC header)/X = [192 + 8× (L+28)/X] µs

T_backoff (assuming single packet in channel) = ½× CWmin× Tslot= 300 *µs*

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, denoted by *T_802.11(N, L, X),* is given by:

$$T\_802.11(N, L, X) = N \times (T\_backoff + DIFS + T\_RTS + SIFS + T\_CTS + SIFS + T\_DATA + SIFS + T\_ACK)$$
$$= N \times [1340 + 8*(L+28)/X] \ \mu s \qquad (1)$$

For DCMA, RTS packets have an additional label field (1 byte) 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 (1byte), and a flag to indicate ACK/RTS (~1 byte). Hence, for DCMA,

T_RTS_DCMA = (T_PHY + 21 bytes @ 2 Mbps) = (192 + 84) =276 µs

T_ACK_RTS = (T_PHY + 14 + 6 + 1 + 1 bytes = 22 bytes @ 2 Mbps) = (192 + 88) =280 µs

Now, consider the case of pipelined transfers from H_1 to H_{N+1} using the DCMA protocol. In this case, the backoff time 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, denoted by *T_DCMA (N, L, X), is given by:*

$$T\_DCMA(N,L,X) = T\_backoff + DIFS + T\_RTS\_DCMA + N\times (3\times SIFS + T\_CTS + T\_DATA + T\_ACK\_RTS)$$
$$= 626 + N \times [750 + 8\times(L+28)/X] \ \mu s \qquad (2)$$

We plot the latency for a 7-hop chain for DCMA and 802.11 with different data rates (2, 5.5 and 11 Mbps) and different packet sizes (80 bytes and 1536 bytes) in Fig 6.

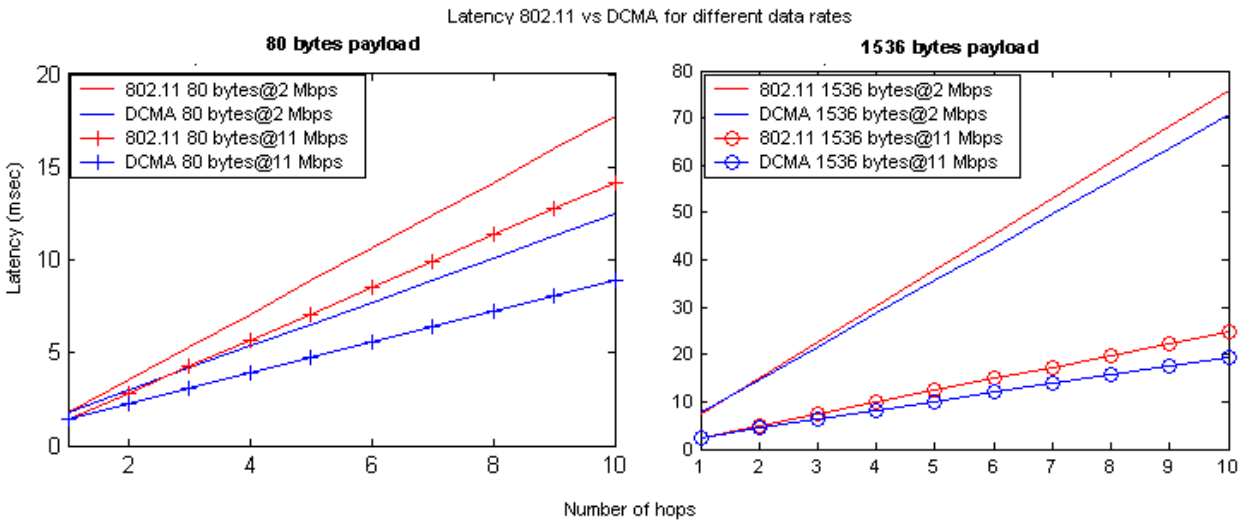


Figure 6 Latency for different data rates and packet sizes (802.11 vs DCMA)

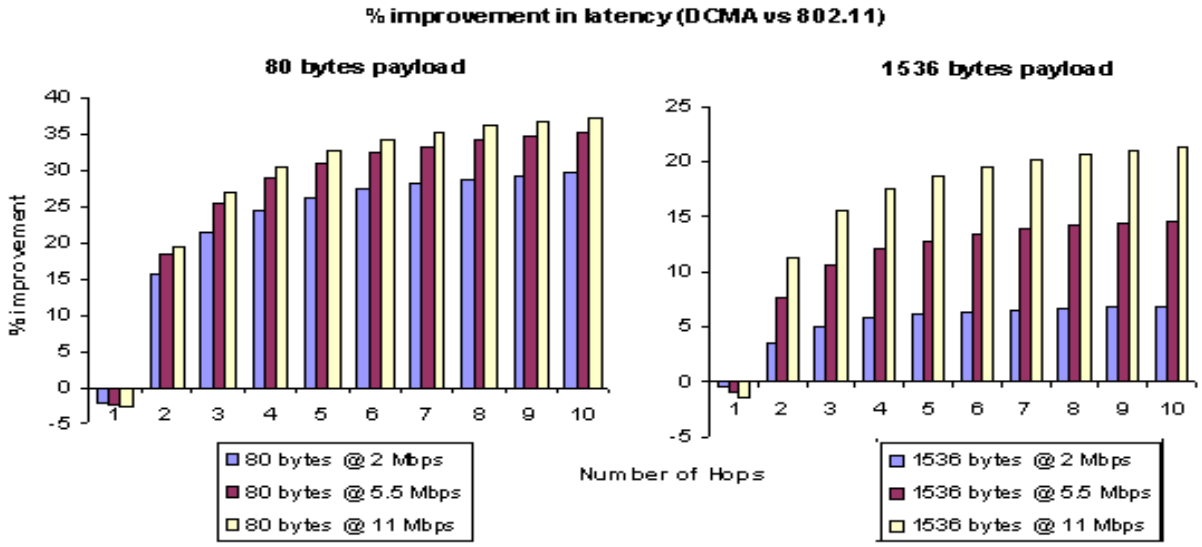


Figure 7 Percentage improvement in latency for different number of hops (DCMA vs 802.11)

We also plot the percentage improvement in latency of DCMA vs 802.11 in Figure 7. The percentage improvement is calculated as [(802.11 latency - DCMA latency)/802.11 latency]×100. As expected, the latency improvements are almost 35% for the smaller packet size (80 bytes) and highest data rate (11 Mbps).

5. SIMULATION RESULTS

To study the performance of our pipelined forwarding mechanism, we implemented the DCMA protocol as part of the ns-2.1b8 simulator [14] with the CMU wireless extensions [15]. This section reports on detailed simulation studies to evaluate the performance of DCMA-based packet forwarding relative to 802.11. As part of our studies, we focus on three metrics: a) the *throughput* improvement achieved by the cut-through protocols and b) the potential reduction in *end-to-end latency* due to the expedited MAC forwarding. 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 throughput was measured by counting the number of received packets at the destination(s). We measured latency only for packets that were received at the receiver. 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 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 tuned to model the Lucent Wavelan card at a 2 Mbps data rate. The effective transmission range is 250 meters, and the interfering range is about 550 meters. All simulated data packets are preceded by an RTS/CTS exchange regardless of the size. Since the data rates for our simulation were constrained to 2 Mbps, it should be clear that our simulation results are *really-conservative*, in that they (almost unfairly) assume the worst-case comparison of DCMA against 802.11. In particular, our performance results are pessimistic, since a) they consider data packets to be sent at the lowest rate, and b) they fail to account for the NIC-to-host packet transfer delays (which is assumed to be zero throughout our simulations), both of which would cause 802.11 to perform much more poorly. We present results for two different topologies:

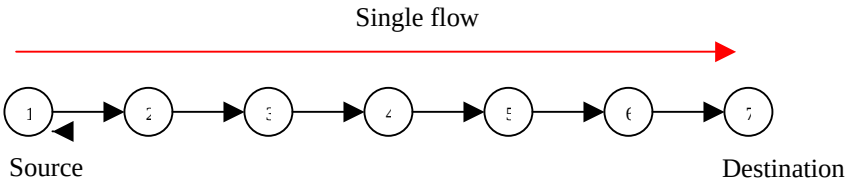
Chain- This topology consists of a set of nodes arranged linearly with a spacing of 248m between nodes. This ensures that the neighbor is just within transmission range of each node (default transmission range for ns-2 is 250m) and the interference range (550m) covers two hop neighbors.

Grid- We used a 10X10 grid with random sources and destinations and varying number of flows.

5.1 Investigations on the Chain Topology

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

A. Single Flow Topology: In this scenario, the traffic consists of a single UDP flow between the two end nodes of the chain. Table 1 lists the simulation parameters used for this topology and illustrates how we vary the traffic rate and packet sizes.



Number of nodes	7 chain
Traffic model	CBR, UDP based single flow (from 1 to 7)
Offered load (kbps)	150-900 Kbps (steps of 50 Kbps)
Packet sizes (bytes)	256,1536

Table 1 Simulation parameters for the simple chain single flow topology

Figure 8a and 8b shows the throughput and delay results for DCMA and 802.11 for two different packet sizes. We see that for either packet size, the DCMA throughput is higher than 802.11. Moreover, as expected, even without considering the additional overhead of NIC-kernel packet transfers, DCMA offers almost a 50% reduction in end-to-end delay, especially at higher offered loads. The throughput graph also shows how the throughput saturates at ~1/4th of the channel capacity—this is an expected result since the large interference range implies that only one of four consecutive links can be active at any time [6].

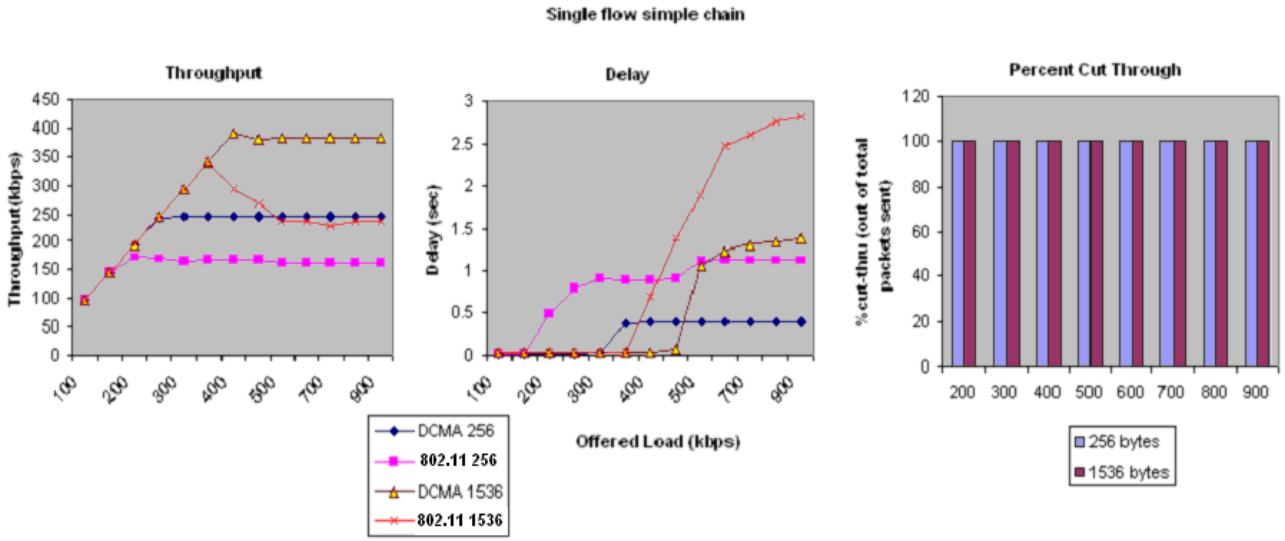


Figure 8a) Throughput, b) Average delays and c) Percent cut-through versus offered load (CBR traffic)

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 ~350 Kbps to ~250 Kbps—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. In this case, as Figure 6 in Section 4.1 shows, the latency difference between 802.l1b and DCMA is not extraordinarily high due to our choice of X=2 Mbps in the simulations; for higher data channel rates, the differences in latency would be more pronounced. Fig. 6c shows the percentage of cut-through packets to the total number of packets delivered from the source to the destination. Since there is only one flow, there is no contention at intermediate nodes to access the channel and hence the percentage of cut-through is 100. We also tested the same scenario with Poisson traffic model instead of CBR. In this case, the traffic was varied from an average offered load of 150

Kbps to 900 Kbps in steps of 50 Kbps. The results for throughput and delay for two different packet sizes are plotted in Fig.9a and 9b respectively show the same trend as in the case of CBR traffic.

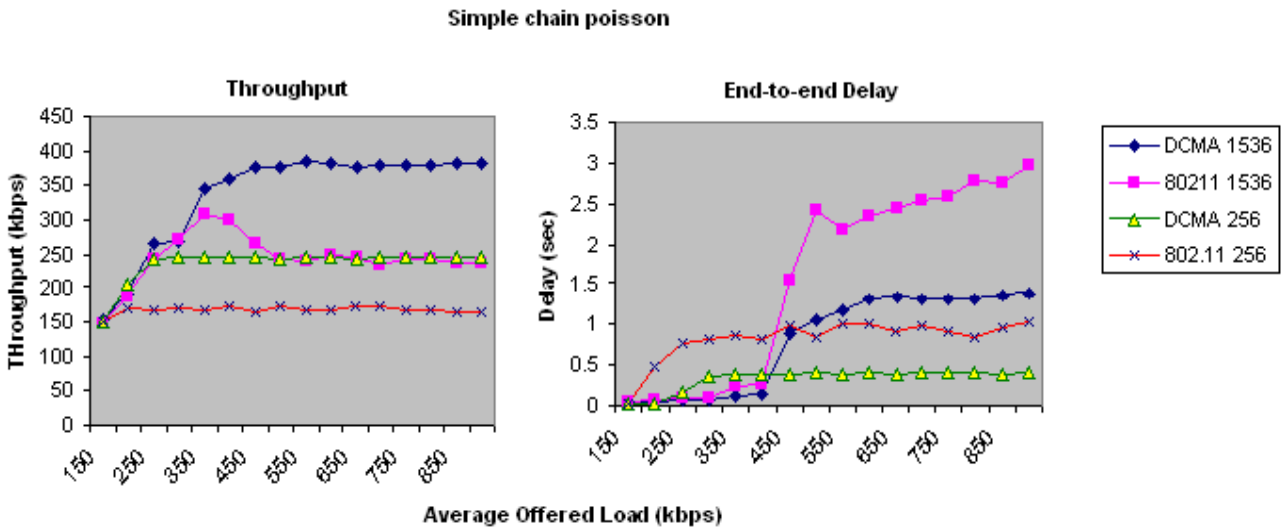
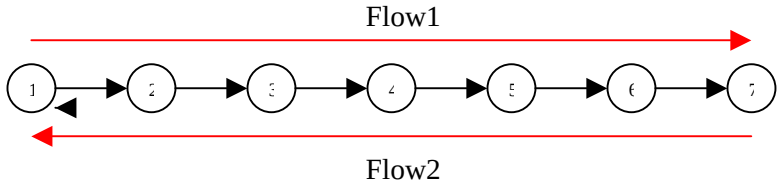


Figure 9a Throughput b) End-to-end delay versus average offered load (Poisson traffic)

B. Simple chain with two flows in reverse direction: To demonstrate the graceful degradation of DCMA to baseline 802.11, we next considered two flows in reverse direction (from node 1 to 7 and from node 7 to 1 respectively). Table 2 explains the simulation parameters for the topology outlined below (we restrict ourselves to 256 byte packets for reasons of space).



Number of nodes	7 (chain)
Traffic model	2 CBR, UDP based flows (1-7 and 7-1)
Offered load per flow(kbps)	150-450 Kbps (steps of 50 Kbps)
Packet sizes (bytes)	256

Table 2 Simulation parameters for the two reverse flows topology

Fig 10 plots the throughput, end-to-end latency and % cut-through rates for DCMA and 802.11 in this case. On again, we see that DCMA is able to obtain significantly higher (almost double the throughput of 802.11) in this case. However, the per-flow delays are not significantly lower, because the presence of two opposing flows implies that either flow will fail to cut-through at some intermediate node. Since DCMA

reverts back to 802.11 channel access at least one intermediate node, the latency differences are not as pronounced for the relatively low X=2 Mbps data channel rate. Note that the two flows have roughly the same throughput implying that each flow gets a fair allocation of the channel for both 802.11 and DCMA.

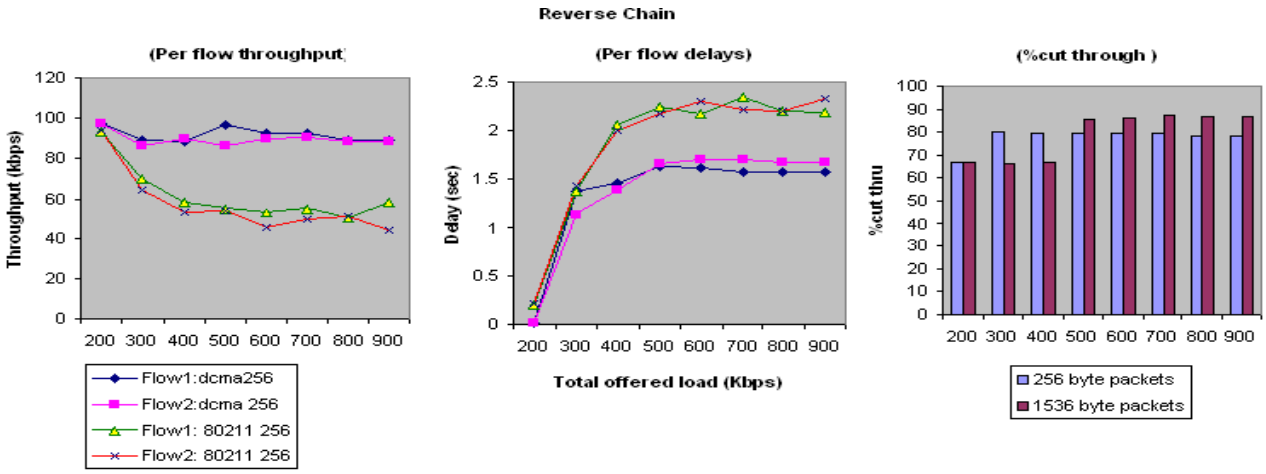


Figure 10a Throughput b) Average delays c) Percent cut-through versus offered load (CBR traffic)

5.2 Investigations With The Grid Topology

After studying the chain topologies described earlier, we tested DCMA on a general 10×10 grid topology (a wireless mesh with 100 nodes) with randomly selected sources and destinations. We generated random source-destination pairs in multiple runs of the simulation and averaged the results obtained for the grid throughput and delays over all the runs. Thus, the grid topology was a combination of the scenarios described earlier (multiple flows-same direction and multiple flows-reverse direction). We showed earlier in the simple chain single flow topology (Fig 8a and 8b) that the performance of DCMA and 802.11 diverges at an offered load of around 300-400 Kbps. This can be attributed to the fact that for low offered loads, the major component of the end-to-end delay is the propagation delay between nodes which is almost the same for both 802.11 and DCMA. At offered loads of about 300-400 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 transmission delays. 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. At very high offered loads, however, a DCMA flow suffers from intra-flow contention, which means the high packet injection rate coupled with cut-

through access causes many more backoffs at the intermediate nodes as compared to 802.11. Based on the above three observation, we expected the best operating range for DCMA to be around offered loads of 300-400 Kbps per flow (for the 2 Mbps channel used in our ns-2 simulations). To test out these hypotheses, we carried out the following set of experiments.

A. Constant total offered load, increasing number of flows

In this case, the total offered load to the grid was kept constant (10 Mbps) across the simulation runs and we varied the number of flows (starting from 10 CBR flows of 1 Mbps each increasing to 40 CBR flows of 250 Kbps each). We ran the simulation for 5 randomly generated topologies and averaged the results for total grid throughput and end-to-end delays, both of which are presented in Figure 11. From the graphs, we can see that when the number of flows is smaller (i.e the individual offered loads per flow are higher), average end-to-end delays for DCMA are actually worse than 802.11. This is because of the higher intra-flow contention as explained before. Also, at lower individual offered loads per flow (40 flows case), the average end-to-end delays for DCMA and 802.11 are comparable. For 20 flows (of 500 Kbps each) and 30 flows (of 333Kbps each), the performance of DCMA is better than 802.11 for throughputs and delays suggesting that this is the desired range of operation for DCMA.

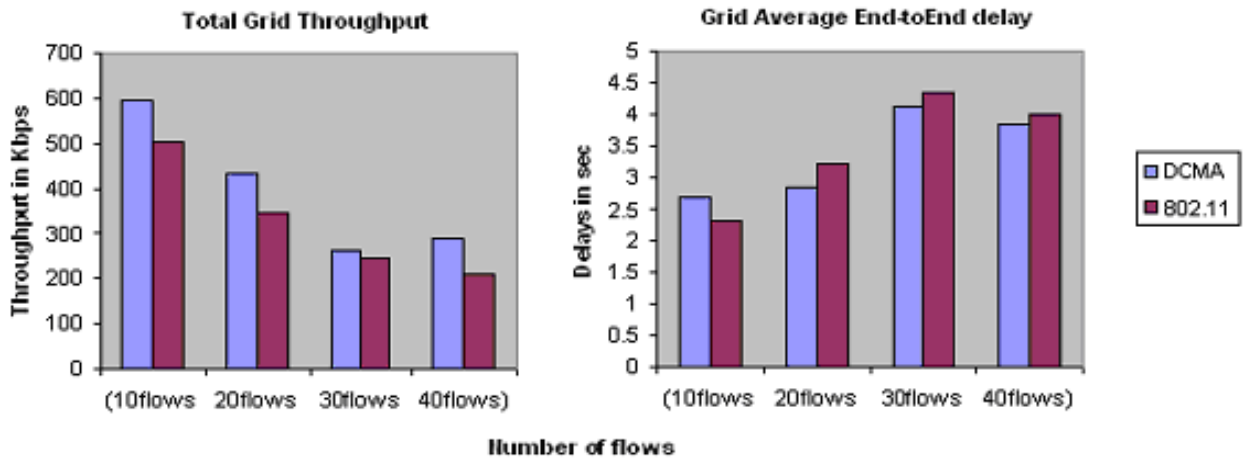


Figure 11a) Total throughputs and b) average delays for 10×10 grid network (Scenario 1)

B. Constant offered load per flow, increasing number of flows

In this case, the individual offered load per flow was kept constant (400Kbps) and the number of flows was increased from 10 to 30 in steps of ten flows. Thus, the total offered load was increased from 4Mbps to12 Mbps. Again, we ran the simulation for 5 randomly generated topologies and averaged the results for total grid throughput and end-to-end delays, which are plotted in Figure 12. We can see that, in this case, DCMA performance for throughput and delay is better than 802.11 (even though, as before, we continue assume that each channel operates only at 2 Mbps). However, as the number of flows becomes higher (30 flows), the probability of successful cut-through decreases, causing DCMA to degrade to the base case of 802.11.

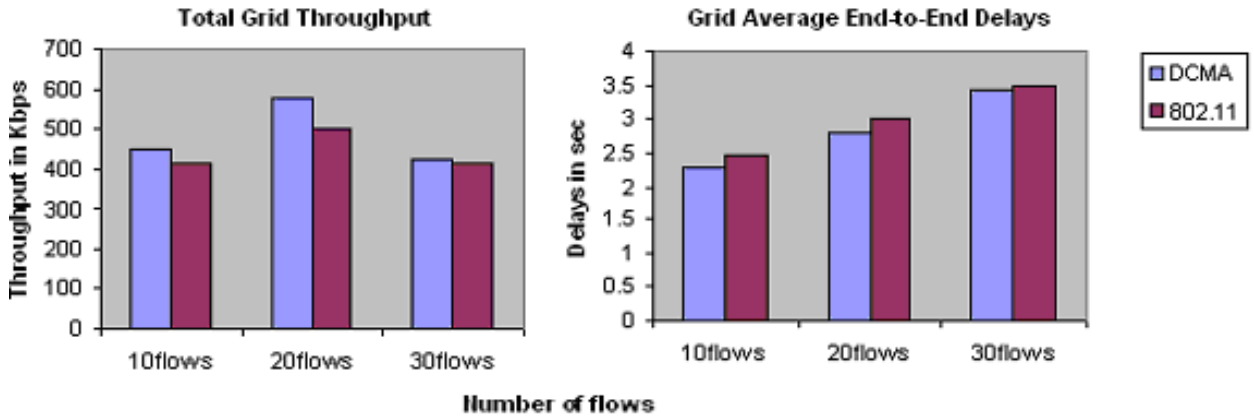


Figure 12 Total throughputs and average delays for 10×10 grid network (Scenario 2)

5.3 Introducing Fairness in DCMA Forwarding

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 is may cause starvation to non-cut-through flows, since the NIC may respond to RTS/ACK requests from an upstream node while it is performing the initial backoff for packets in the packet queue of its own host. As a simple fix to this problem, we modified the interface behavior to essentially monitor whether the host packet buffer was empty or not. If the host had any packets pending in its MAC buffer, then the NIC would not accept (decline to send back a CTS) an RTS/ACK request from a neighbor attempting to perform cut-through forwarding. Under this approach, cut-through would be restricted only to situations where the overall load was low enough (or the traffic arrival rate was intermittent enough) to ensure that many intermediate nodes in the forwarding

path had no packets of their own to transmit. 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). Figs.13a and 13b show per flow throughputs and delays before and after applying the fairness policy for two flows on a simple chain topology (node 1-7 and node 4–7). We can see that with fair-DCMA individual flow throughputs and delays are much closer than DCMA without fairness. Note that other fairness policies may also be applied; however we defer the discussion to a future paper.

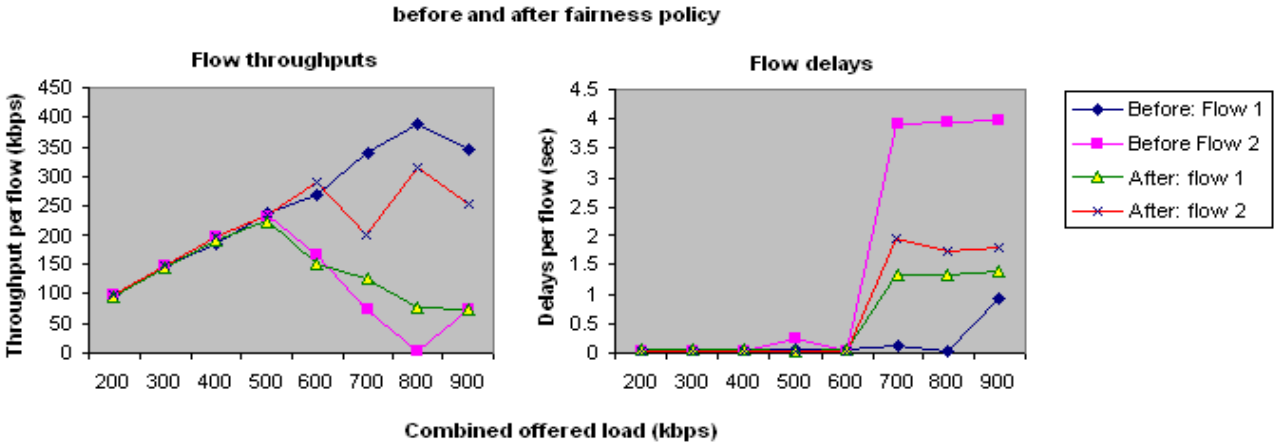


Figure 13 a and b) Per-flow throughputs and delays (for DCMA) before and after fairness policy

6. CONCLUSIONS

In this paper, we presented an efficient 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. This was made possible by enhancing the 802.11 DCF channel access scheme and by carrying a label in the RTS/ACK packet, which allowed the NIC to determine the packet's next hop. The NIC was augmented with a label-switching table mapping incoming labels and MAC addresses to outgoing labels and MAC addresses.

Extensive simulation studies show that DCMA performs better than 802.11 in most scenarios. We also identified an important unfairness issue associated with our approach that causes one flow to be starved at the

expense of cut-through access for the other flow. We have also proposed a fairness scheme that uses queue occupancy information at the MAC layer to allow competing flows to get fair shares of channel to send their traffic. Using simulation results, we have shown that the improvement in fairness amongst flows by employing the fairness scheme at intermediate nodes. We should note again that, in practice, the gains are likely to be much higher, since we neglected both the NIC-to-host packet transfer overheads and ran the experiments on the slow 2 Mbps data channel. Also, using the results of the grid experiment, we have identified a useful operating range for using our algorithm to achieve a better throughput and delay performance as compared to 802.11. In general, packet cut-through is especially useful at relatively low to moderate network loads, where many hops essentially lie idle. *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.*

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] IEEE Computer Society. 802.11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 1997.

[2] B. Crow, I. Widjaja, J. G. Kim and P. T. Sakai, "IEEE 802.11 Wireless Local Area Networks", *IEEE Communications Magazine*, Sept '99.

[3] Community Wireless/Rooftop Systems, www.practicallynetworked.com/tools/wirelessarticles_community.htm

[4] R. Karrer, A. Sabharwal and E. Knightly, "Enabling Large-Scale Wireless Broadband: The Case for TAPs", *HotNets-II*, November 2003.

[5] Xu. S.G. and Saadawi T. "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?", *IEEE Communications Magazine. 39(6): 130-137*, June 2001.

[6] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee, R. Morris, "Capacity of Ad hoc Wireless Networks", *Proceedings of ACM International Conference on Mobile Computing and Networking*, August 2001

[7] M. Gerla, K. Tang and R. Bagrodia, "TCP Performance in Wireless Multi-hop network", *IEEE WMCSA99*, Feb '99.

[8] Jungmin So and Nitin Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver", *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004.

[9] MPLS label switching architecture, IETF RFC 3031

[10] LDP Specification , IETF RFC 3036

[11] D. Johnson and D. Maltz., "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing, chapter 5*, Kluwer Academic Publishers, 1996.

[12] C. Perkins, E. belding-Royer and S. Das., "Ad-Hoc On-Demand Distance Vector (AODV) Routing", *draft-ietf-manet-aodv-09.txt, IETF*, Work in Progress, November 2001.

[13] B. Jabbari, R. Papneja and E. Dinan, "Label Switched Packet Transfer for Wireless Cellular Networks", Proceedings *of IEEE WCNC*, August 2000.

[14] Kevin Fall and Kannan Varadhan, *ns* Notes and Documentation. Technical Report, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997.

[15] CMU Monarch Group extensions to *ns*, http://www.monarch.cs.cmu.edu/.

[16] A. Acharya, A. Misra, S. Bansal, "A Label-Switching Packet Forwarding Architecture for Multi-Hop Wireless LANs", *Proceedings of the Fifth International Workshop on Wireless Mobile Multimedia, ACM WoWMoM 2002*, Atlanta, Georgia.