

BBN Technical Memorandum No. TM-2023

Ultra Low Latency MANETs

Job Number: 13234001

September 25, 2006

Prepared for:

DARPA
3701 North Fairfax Drive
Arlington, VA 22203

Prepared by:

BBN Technologies
10 Moulton Street
Cambridge, MA 02138

Approved for Public Release. Distribution Unlimited

Ultra Low Latency MANETs

Ram Ramanathan
Internetwork Research Department
BBN Technologies
Cambridge, MA 02138
Email: ramanath@bbn.com

Fabrice Tchakountio
Mobile Networking Systems Department
BBN Technologies
Cambridge, MA 02138
Email: ftchakou@bbn.com

Abstract— While the *capacity* of Mobile Ad Hoc Networks (MANETs) has received considerable attention, the issue of end-to-end *latency* has been largely ignored. As MANETs get larger and denser with an increasing number of end-to-end hops, the latency and jitter experienced by packets will be prohibitively high for existing and emerging real-time applications. We present and evaluate a design concept for achieving an order-of-magnitude latency reduction in large-scale MANETs. Our approach includes two key components: a *relay oriented physical layer* that uses cut-through relaying by pipelining the bits through the receive and transmit chains; and a *path access control* mechanism for reserving the floor multiple hops at a time. We present a rough numerical analysis of the relative performance gain from our ideas, complemented by a simulation-based analysis. Our study shows that, depending upon the parameters used, a 4x - 10x reduction in latency over legacy systems is possible using the mechanisms described in this paper. Finally, we present some thoughts on implementation of our proposed system.

I. INTRODUCTION

In a *mobile ad hoc network* (MANET)¹ a packet typically travels over several *hops* enroute to the final destination. Each hop consists of a *relay* node receiving the packet, processing it, queueing it, deciding the next hop, re-contending for the channel and finally retransmitting the packet. Thus, the packet experiences a number of delays at each hop: the per-hop transmission delay, the processing delay, the channel access delay and the propagation delay. While the sum of these delays is not a problem for small-diameter networks, they are significant enough to pose a serious problem to scaling a MANET to a large number of hops.

MANETs are trending toward larger numbers of nodes due to increasing demand and ever-decreasing radio cost. They are also trending toward shorter range hops for a number of reasons. First, higher capacity (data rate) implies a higher frequency band because that is where more spectrum is available. However, propagation characteristics are poorer at

higher frequencies, implying short ranges. Second, higher data rates are obtained by higher modulation schemes which require higher SNR, which in turn point to short ranges. Finally, many MANETs need to be energy conserving which again implies shorter ranges. We are already seeing the trend in 802.11 which moved from 900 MHz in the mid-90's to 2.4 GHz and now increasing presence in the 5.8 GHz ISM band. Increasing network size and shortening communication ranges foretell MANETs with very large diameters. Indeed, an architecture where dense low-cost relays self-form into a topology consisting of short-range high-data-rate links is emerging as a blueprint for next generation MANETs, and for example, is the basis of the visions in [1], [2].

Limiting the end-to-end latency and jitter is of paramount importance in a number of real-time applications such as high-definition interactive video, packetized voice, real-time imaging, and advanced distributed virtual-reality applications such as telemedicine. In order for these applications to seamlessly extend to wireless MANETs, we need to provide comparable latency and jitter performance – something possible today only with small MANETs. While the capacity challenges of these applications have been much researched, the latency challenges remain unsolved.

Given the topological trends in MANETs and the tightening latency requirements of emerging applications, a reasonable goal for MANET research in the next 3-5 years is a 10000 node network with at most 100 msec roundtrip delays. Using a Manhattan grid as a rough approximation, the diameter of such a network is about 140 hops (hypotenuse of a 100x100 grid) giving an average per hop latency goal of about 350 usec. Latency in current MANETs is an order of magnitude higher than this, typically

¹The term is used here to mean peer-peer, self-organizing, wireless, possibly mobile, multihop networks, and broadly includes *mesh networks*, *sensor networks* and *military packet radio networks*

about 8 ms per hop² even under light load. What is more, even if we hypothetically eliminated all delays except transmission/ reception delay it would still take 400 usec per hop and by shy of our goal! In other words, improving efficiency by nifty MAC layer protocols and good engineering helps but will not even come close. There is a need for a radical rethink of multi-hop relaying.

We present a design concept for *ultra low latency* MANETs for an order-of-magnitude latency reduction in large-scale MANETs. We re-examine two basic tenets of current day MANET thinking, namely, the need to re-contend at every hop, and the need to first receive before transmitting. We show that both of these needs can be relaxed considerably, enabling a new approach that pushes latency to a new performance plane altogether. Our design includes two key components: a *relay oriented physical layer* (ROPL) that uses cut-through relaying by pipelining the bits through the receive and transmit chains; and a *path access control* (PAC) mechanism for reserving the floor multiple hops at a time. Packets departing the source are switched at the physical layer, without re-contention, until they encounter an ongoing flow at which point that particular *segment* is terminated and a new segment is initiated, and so on until the destination is reached.

We make two architectural shifts in our solution for ultra low latency: access control is no longer a purely local mechanism and has semi-global scope; switching/forwarding of a packet is no longer done at the network layer, it is done at the physical layer, and is cut-through. The PAC reduces the average re-contention and switching delay, whereas the ROPL reduces the effective per-hop transmission time to the length of the packet header rather than the entire packet. This concept was initially introduced in [4]. In this paper, we take it to the next several levels of detail, design and analyze our mechanism.

We analyze the performance of our proposed design in two ways. We present a numerical analysis of the latency reduction using PAC. We also present a simulation-based study of the latency reduction using NS-2 on stationary networks. Finally, since our approach involves modifying the physical layer,

²Based on [3], our own experience, commercial mesh network specifications. Assumes data rate of 10 Mbps, packet size of 500 bytes (see section III for a rationale of these numbers)

a natural question is that of practicality. We argue that the use of software radios eliminates many of the common hurdles with proprietary hardware and present an implementation sketch based on the open source GNU software radio that we are enhancing as part of the DARPA ACERT program [5].

Ultra low latency MANETs would be of interest in a number of future scenarios: a rooftop based community or municipal mesh network with one backhaul node every 10,000 or so homes (to reduce cost, which is currently an issue); future military networks of robots, vehicles and soldiers [1]; large-scale sensor networks that are also used for remote controlling mobile sensors, etc. It would also be an enabling technology for visions such as Smart-Dust [2]. While motivating the need is important, we also note that applications have a tendency to emerge once the technology is there (“build it and they will come”).

To summarize, our contributions are: the first approach to MANET delay reduction that goes beyond just switching time reduction; a viable design for cut-through routing at the physical layer; theoretical and experimental analysis of the relative performance of our design.

The rest of the paper is organized as follows. In section II we survey prior work in this area. In section III, we describe the basic architecture, our relay oriented physical layer design, and two mechanisms for path access control, along with a numerical analysis of relative performance. In section IV, we present a simulation-based analysis of path access control over relay oriented physical layer, and in section V

II. RELATED WORK

The idea of switching at the physical layer immediately reminds one of “cut-through” or “wormhole” routing techniques in wireline networks [7], [8]. In cut-through routing, packets entering a network node on one interface are forwarded, without storing, on another interface. Although the basic idea is similar, the problem is vastly different in ad hoc networks, chiefly due to the *broadcast nature* of communications and the resultant need for access control. Further, transmission and reception require a number of steps in waveform processing, and node mobility requires dynamic switching (transit table).

Medium access control in ad hoc network has been a much-studied topic. The research is motivated by the fact that standards designed for wireless LANs (such as IEEE 802.11 [6]) do not work well in ad hoc networks. These and most research in this area attempt to increase the MAC efficiency by allowing as many simultaneous transmissions as possible. While increasing efficiency reduces channel access delay, most of these techniques are *per hop* techniques in the sense that packet has to re-contend all over again after being relayed one hop, thereby increasing the latency. In contrast, we use *path oriented* access where a packet, acquires the channel for multiple hops at a time so that there is no recontention delay.

In the context of mobile ad hoc wireless networks, there exists work on “label switching” at the medium access or link layer [9], [10] which pushes the forwarding function one layer down. In [9], access time is reduced by having the ACK for a packet double up as an RTS for the next hop. Our architecture involves a more fundamental change compared to these works – switching is done not at the MAC but the physical layer, and floor acquisition is done for multiple hops at a time. Unlike the other works, our architecture allows sending a part of the packet while receiving another part - this is key in bringing down the latencies so as to meet the stated goals.

III. SYSTEM DESCRIPTION

In this section, we describe the design of a system for ultra-low-latency packet transport in MANETs. We begin by describing our proposed system architecture that hosts the Path Access Control (PAC) and the Relay-oriented Physical Layer (ROPL). The basic ideas underlying this system concept were first presented in [4], but without sufficient detail. Here we take those high level ideas and flesh out the details.

A node in our system consists of a single transmitter and a single receiver, each consisting of the RF front end and modem functions. These are independently and quickly tunable to one out of k mutually orthogonal *channels*³, and have separate

³In the current design, a channel is a frequency but our approach can work equally well if channels based on orthogonal codes are used

antennas. Thus, full-duplex operation (transmitting while receiving) is possible.

Packets go from a source s to a destination d in units of *segments*. A segment (r_0, r_t) is a sequence of nodes $r_0, r_1, r_2, \dots, r_t, t > 0$, such that at each relay node $r_i, 1 < i < t$, the packet is cut-through switched at the physical layer without going to the PAC or higher layers. At r_t , the segment terminates, that is, the packet is sent up to the PAC. Peer PAC processes at the two ends of the segment (r_0 and r_t) communicate to implement the segment. The PACs in the intermediate relay nodes are *not* involved in the transport of a packet over a segment. We shall refer to the node at which a segment begins (e.g. r_0) as the *Segment Start Node (SSN)* and the node at which a segment ends (e.g. r_t) as *Segment End Node (SEN)*.

Note that, as shown in Figure 1, unlike conventional architectures where the packet has to go up three layers and down three layers at each hop, our system only involves the physical layer most of the time, and even at SENs requires going up only two layers. Another difference is that routing control (topology updates, route requests, Hellos etc.) is a cross-layer function, with the forwarding table being stored at the physical layer.

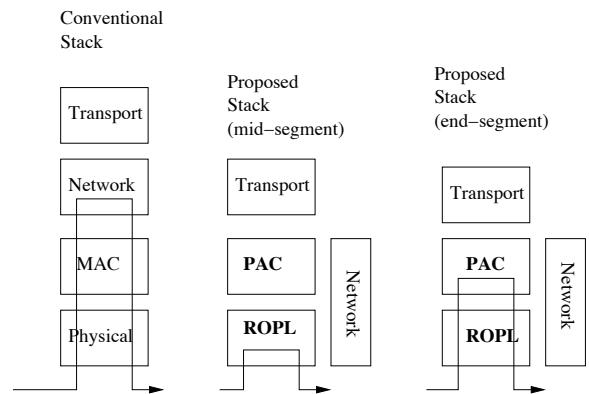


Fig. 1. Conventional layering and data path (left), our layering in the intermediate nodes (middle), and our layering at segment ends (right).

The PAC at the head of a potential segment seeks to make the segment as long as possible, ideally all the way to the destination. The longer the segment, the less the number of times the packet has to exit the physical layer. However, a segment may need to be terminated for several reasons: there may be an ongoing “blocking” flow that doesn’t allow floor acquisition after some hops; mobility

or fading might have broken the chosen path; or the segment length may have exceeded a maximum dictated by the source (estimated based on, say, expected error accumulation, more on this later), etc. When a segment is terminated, the packet goes up to PAC layer which initiates a new segment. Thus the minimum number of segments in the path from S to D is one, and the maximum number is the hop-length of the path.

A number of solutions are possible for implementing the concepts of path-oriented medium access and cut-through relaying. This paper presents an initial set of mechanisms that work together to achieve ultra-low-latency transport. These mechanisms will be described in detail in the forthcoming sections – here we briefly overview their operation.

The ROPL works as follows. As soon as a node receives the preamble and the first few bytes of a packet, it uses information this (physical layer header) to determine whether to relay, discard, or keep the packet. The routing protocol, which could be any scalable table-driven protocol (e.g. [11], [12], provides cross-layer topological information to help make this decision. If the decision is to relay the packet, and this node is not the end of a segment, then the ensuing bits are pipelined out through the transmit chain. That is, the incoming bit stream goes through the receive chain and is demodulated, decoded, despread if necessary just as in conventional receivers. Immediately, and without waiting for the rest of the packet to arrive, the received bits are sent to the transmit chain and transmitted on a channel different from the one it was received in.

For the ROPL cut-through pipelining to work, the relay node must have secured access to the channel prior to receiving the packet. We contend that access has to *path oriented*, reserving the floor for multiple hops at a time. We have developed two approaches for path access control: *simple PAC (S-PAC)* and *tailgating PAC (T-PAC)*. Both are based on extending the CSMA/CA philosophy to multiple hops. In S-PAC, the floor is reserved for the entire segment before the DATA packet leaves the source. In T-PAC, the DATA packet trails (“tailgates”) the floor acquisition so that the floor is acquired “just in time”. Simple PAC is easier to understand and implement but wastes capacity and has higher latency.

Tailgating PAC is more efficient. Both are multi-channel mechanisms in that the control and DATA packets may use different channels at each hop. The PAC ensures that the channels are chosen so that full-duplex pipelining (transmitting part of the packet while receiving another part) is enabled and at the same time inter-flow interference is avoided. This allows ROPL to do its job without worrying about next hop access.

The next two subsections describe our design in detail. Along with the descriptions, we also provide some quantitative insights based on rough numerical analysis on simple network topologies (line, manhattan grid, etc.).

A. Relay-Oriented Physical Layer

Our design concept for a Relay Oriented Physical Layer (ROPL) is shown in Figure 2. The key idea is to couple the *receive chain* of a physical layer to the *transmit chain* and control the coupling so that the node can, if necessary, transmit the one part of the packet while a later part is being received. Such a coupling conventionally occurs at the network layer. By doing it at the physical layer, we are able to pipeline the bits, dramatically decreasing the switching latency.

The ROPL leaves the receive chain and the transmit chain themselves completely unchanged. Thus, the ROPL inherits the properties of the conventional physical layer in terms of transmit and receive capabilities, including data rate, error recovery, spreading etc. A few simple processing blocks are added on the “relay path” within the physical layer, as shown in the figure and explained below.

The *transit buffer* is a conflict-protected circular buffer that stores the incoming bit stream and allows simultaneous read/write so that the transmit chain can access and transmit the bits. The *transit control* is a key part of our design and contains the logic for deciding the setting of the *switch* – whether to drop, keep, or relay the packet. It makes this decision using the received packet header and using it in conjunction with the *relay table* and the *floor state*. The relay table contains the next hop information for each destination and is populated in a cross-layer fashion by the routing protocol. It also decides whether to pipeline the incoming bits or not using a “mode” bit in the header (see below).

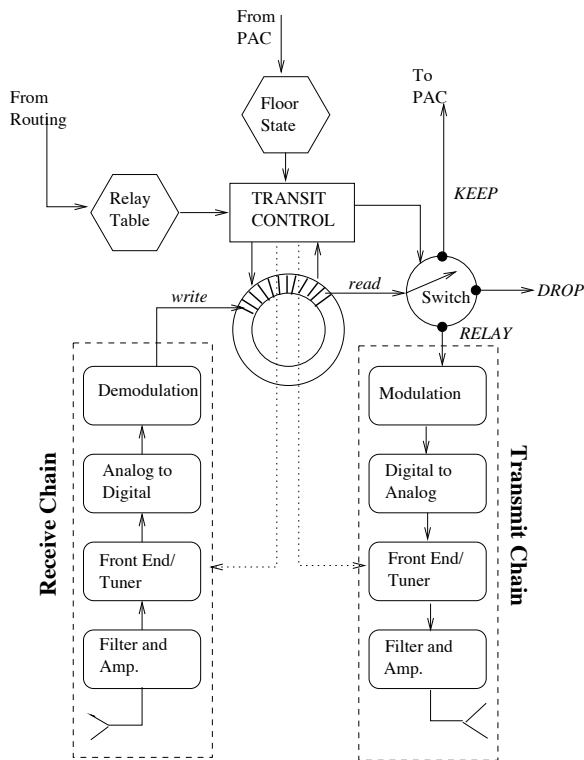


Fig. 2. Block diagram of our Relay Oriented Physical Layer design concept. The new components are the ones outside of the Transmit and Receive chains.

The *floor state* is populated by the PAC and says whether or not the floor is currently “acquired” for transmission. For example, if a neighboring node has an ongoing flow, then the PAC, using techniques described in section III-B, marks the floor as busy for a certain duration. In such a case, the node cannot further relay the packet and delivers it to the PAC.

We note that all of the new components we have added pertain to processing logic, table look up or buffer management, and nothing to do with signal processing. As such, this new functionality may be easily implemented as an add on FPGA or if software radios are used, as straightforward additional lines of code. We discuss the latter approach in section V.

The physical layer header format is inherited from the 802.11 DSSS Physical Layer Convergence Procedure (PLCP), with three new fields added, as shown in Figure 3⁴. The new fields are shown in bold in the figure and are as follows.

The *mode* field indicates whether the packet is to

⁴We have chosen 802.11 because it is the most commonly available chipset, but the same enhancements can certainly be done to most other physical layers.

Sync (128 bits)	SFD (16)	Signal (8)	Service (8)	Mode (2)	Dest (48)	Next Hop (48)	Length (16)	CRC (16)
--------------------	-------------	---------------	----------------	---------------------	----------------------	--------------------------	----------------	-------------

Fig. 3. Relay Oriented Physical Layer header

be pipelined or not. The *destination* field contains the address of the final destination of the packet. The *next hop* field contains the the address of the one-hop reachable node for which this packet is intended. The other field semantics are exactly as in 802.11. Briefly, the *sync* and *SFD* comprise the preamble, the *signal* field indicates the data rate used (1, 2, 5.5 or 11 Mbps), the *service* field is unused, the *length* is the number of microseconds needed to transmit the payload, and *CRC* is the error detecting check on the header (which will now be over the three new fields). We refer the reader to [6] for details.

When a packet synchronization is acquired and the header is completed in the transit buffer, first the *mode* field is examined. If the mode is non-pipelined, the rest of the packet is received and handed over to PAC for processing. If mode is pipelined, then the *next hop* field is examined. If the next hop is not equal to this node, the switch is set to “keep” the packet and send it up to the PAC⁵, otherwise the *destination* field is used to perform a lookup in the relay table. The result of the lookup is either a “keep” (when this node is the segment end) or “relay”. In the “keep” case, the packet is delivered to PAC.

In the “relay” case, the “floor state” is checked. If it is BUSY, the packet is delivered to PAC. Otherwise, the ROPL rewrites the next hop field of the header (which resides at this time in the transit buffer) and sets the switch to connect to the transmit chain. This happens concurrently with the reception of the remainder of the packet, and as soon as this happens, the transmit chain may (also concurrently) start reading and transmitting the packet.

When the PAC originates a packet, as it will if this node is an SSN, the ROPL header fields such as destination and next hop are left unfilled to honor layering. The ROPL puts the packet in the transit buffer, fills the destination field using the call

⁵Why not discard? Because the DATA packet could have some information in the header needed by PAC – e.g. if this is part of a burst, and more DATA packets are to come, this is indicated in the header (see section III-B for more detail)

parameter and the next hop field using the relay table. It then sets the switch to relay and gets the transmit chain to send the bits over the air.

A key issue in pipelining is the need for orthogonal (frequency) channels. The frequencies must be far enough apart and there must be enough isolation between the transmit and receive chains so that there is no “leakage” that interferes with the reception. This is easier to do in higher frequency bands. We favor the use of the 5.8 GHz U-NII band (IEEE 802.11a operates in this band) which has a large number of channels and a wide enough spread between them to enable orthogonality.

What kind of relay latency improvement does the ROPL give? To get a rough idea, we present a back-of-the-envelope analysis. Let P be the payload size (excluding any protocol headers) in bytes, and r be the data rate in bits/usec (which is same as Mbps). Suppose this is a voice packet and uses UDP/RTP/IP/802.11, which implies a header size of 98 bytes (40 bytes UDP/RTP/IP, 34 bytes MAC header and 24 bytes PLCP header). We assume that the total time to process a frame is approximately the same with and without ROPL (in reality ROPL would likely be faster since the packet does not have to leave the physical layer), and is 5 usec⁶. Then, conventional switching latency is

$$\lambda_{conv} = \frac{8 \cdot (98 + P)}{r} + 5usec \quad (1)$$

With ROPL, we only have to receive the ROPL header and process it, rather than the entire packet. From Figure 3, the size of the ROPL header is 290 bits. Thus, the ROPL switching latency is

$$\lambda_{ropl} = \frac{290}{r} + 5usec \quad (2)$$

Then, the ratio between the latencies of conventional switching and ROPL is given by

$$\Gamma_{ropl}^{conv} = \frac{(98 + P) \cdot 8 + 5r}{290 + 5r} \quad (3)$$

Figure 4 plots Γ_{ropl}^{conv} , referred to as the *latency reduction factor* as a function of the data rate for different values of P .

⁶This is likely to be a gross overestimate. We use this, conservatively, because we could not find any work that measures this in current hardware. Using a lower number would only increase the absolute and relative performance of ROPL.

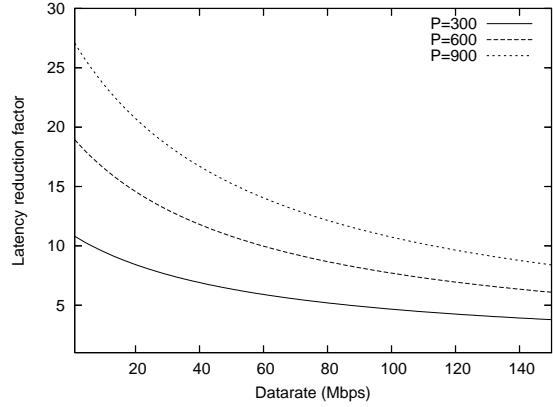


Fig. 4. Latency reduction factor of ROPL switching over conventional switching

We observe that at today’s data rates, ROPL reduces latency well in excess of a factor of 10. As the data rate increases, the latency reduction factor reduces because the processing/switching latency of 5 usec starts to dominate the transmit delay. Nonetheless, even at very high data rates, there is a 400% to 800% improvement in switching latency. This is actually very conservative because we have not factored in the decrease in processing speed that is inevitable. Also, clearly the gain is more for larger packets. Another way of looking at it is that ROPL allows us to use larger packets and still meet latency requirements while cutting down on the header overhead.

B. Simple PAC

The Simple PAC (S-PAC) is an extension of the traditional CSMA/CA mechanism to a segment-oriented regime. That is, everything that happened over a hop – control packets, floor reservation, error detection, re-contention – now happens over a segment. Recall that CSMA/CA consists of a Request-To-Send (RTS), Clear-To-Send (CTS), DATA, and ACK handshake between the sender and receiver over one hop. Nodes that hear the RTS and CTS defer using a Node Allocation Vector (NAV), thereby enabling the communicating pair to acquire the floor for the duration of the handshake. In S-PAC, the handshake occurs between the Segment Start Node (SSN) and the Segment End Node (SEN), with the intermediate nodes behaving merely as “re-energizers” of the packet by performing cut-through relaying. In the case of a single-segment path, the

SSN is the packet source (originator) and the SEN is the packet end-destination. We shall refer to the control packets in S-PAC as S-RTS, S-CTS, and S-ACK (indicating “segment RTS etc”). An additional packet S-TRM (Segment Termination) is used for terminating segments (see below).

The S-PAC allows for multiple DATA-ACK packets within a single RTS-CTS handshake. This is because ultra low latency is motivated real-time applications such as voice and video and most such applications generate bursts of back-to-back packets. It makes sense to flush all packets queued to a destination rather than do an RTS-CTS every single packet. This facility has been proposed for hop-based conventional 802.11 in [13] by making use of the “More Fragments” bit. We have used this bit in a similar manner. Further, we assume that the conventional baseline for comparison purposes is this enhanced burst mode MAC.

The S-PAC uses four orthogonal frequency channels, named 0, 1, 2 and 3 for description purposes (in an implementation these numbers can serve as an index into actual frequencies). One channel (channel 3) is the control channel and carries the S-RTS, S-CTS, S-TRM and S-ACK. The other three channels are DATA channels. The control packets are all relayed using the non-pipelined mode of the ROPL (see section III-A) whereas the DATA packets use the pipelined mode. When idle, nodes are tuned to the control channel.

The packet formats for S-RTS, S-CTS, S-ACK and S-TRM are shown in Figure 5 and explained below

S-RTS

Frame Cntrl (16 bits)	Max Seg (16 bits)	Hop count (16 bits)	Relay Address (48 bits)	Source Address (48 bits)	CRC (32 bits)
--------------------------	----------------------	------------------------	----------------------------	-----------------------------	------------------

S-CTS

Frame Cntrl (16 bits)	Duration (16 bits)	Source Address (48 bits)	CRC (32 bits)
--------------------------	-----------------------	-----------------------------	------------------

S-DATA

Frame Cntrl (16 bits)	Source Address (48 bits)	DATA PAYLOAD (up to 2400 bits)	CRC (32 bits)
--------------------------	-----------------------------	-----------------------------------	------------------

S-ACK

Frame Cntrl (16 bits)	Source Address (48 bits)	CRC (32 bits)
--------------------------	-----------------------------	------------------

S-TRM

Frame Cntrl (16 bits)	Transmitter Address (48 bits)	Receiver Address (48 bits)	CRC (32 bits)
--------------------------	----------------------------------	-------------------------------	------------------

Fig. 5. Control and data packet headers for Simple Path Access Control

First, the *Frame Control* and *CRC* fields are similar to those in 802.11. The basic function of the frame control field is to indicate the control packet type, and the CRC is used to detect errors.

In the S-RTS, the *Max Seg* is the maximum length of the initiated segment and is set by the SSN when it originates an S-RTS. The *Hop count* field indicates the number of hops the S-RTS has traveled thus far and is incremented by each relay node. The *Relay Address* is the address of the intermediate relaying node transmitting the S-RTS. The *Source Address* is the address of the originator of the packet (the SSN).

In the S-CTS, the *Duration* field indicates the maximum duration from current time that the handshake will be active. It is calculated and set by the SEN. The *Source Address* here is the address of the packet destination (SEN), which originates the S-CTS.

In the S-DATA, the *Source Address* is again the SSN, and the DATA is the payload. In the S-ACK, the *Source Address* is the address of the SEN. In the S-TRM, the *Transmitter Address* is the address of the node sending the S-TRM, and the *Receiver Address* is the address of the node for whom the S-TRM is intended. The S-TRM is a one hop control packet.

We note that the neither the end-destination nor the immediate (next hop) receiver address are indicated in the above packets. This is because they are part of the ROPL header.

We now describe the S-PAC operation in terms of the steps involved in sending a packet from a source S to a destination D in a connected network.

- 1) S becomes the *Segment Start Node* (SSN) and initiates a segment.
- 2) The SSN sends an S-RTS with a configured “maximum segment”, initializes hop count to zero, and sets the relay and source addresses to its address. It includes the destination and next hop (R) in the ROPL header. The S-RTS is transmitted on the control channel.
- 3) Node R processes the S-RTS in non-pipeline mode as follows. If the hop count is within the maximum segment size, and the floor state is not BUSY, it does the following:
 - Increment the hop count in the S-RTS.
 - Replace the relay address field in the S-

RTS with R

- Retransmit the S-RTS by giving it to the ROPL. Recall from section III-A that the ROPL fills out the next hop field in the ROPL header before transmitting.

Otherwise, it terminates this segment. It constructs an S-TRM packet with the TA set to this node (R) and RA equal to the relay address field. Thus, the S-TRM is addressed to the previous hop on the path.

- 4) Nodes other than R that receive the S-RTS process it as follows. They set their floor state to BUSY. These nodes are called *sentinels* for the S - D flow as they protect that flow by refusing to forward other S-RTS's.
- 5) The S-RTS continues to be relayed until it either reaches the destination, the maximum segment length is exceeded, or it hits a busy node. In the first two cases the current node becomes the *Segment End Node* (SEN) and the last case the previous hop is made the SEN by virtue of the S-TRM.
- 6) The SEN constructs an S-CTS with itself as the source address, and puts the SSN as the destination in the ROPL. It also enters the Duration field as $L_s * (T_{Scts}^h + T_{Sdata}^h + T_{Sack}^h)$ where L_s is the length of the current segment, taken from the hop count field of the S-RTS, and T_{Scts}^h , T_{Sdata}^h and T_{Sack}^h are the per hop delays of the S-CTS, S-DATA and S-ACK packets respectively. This captures the total duration for which the packet transfer will be ongoing relative to the time at the SEN. It also initializes the frequency field to 0, and tunes its receiver to F_0
- 7) The S-CTS is relayed back to SEN in non-pipelined mode. At each relay node, the transmitter is tuned to the frequency F_i in the received S-CTS, the receiver is tuned to $F_{(i+1) \bmod 3}$ and the frequency field is updated to $F_{(i+1) \bmod 3}$. We assume that the S-CTS is relayed back along the same path of relay nodes as the S-RTS⁷
- 8) All nodes receiving the S-CTS set their NAV

⁷While this is typically the case when symmetric links or shortest hops are used, it is not a crucial assumption. The S-CTS can be made to "reverse the route" of the S-RTS by having each relay node pick as the next hop of the S-CTS the previous hop in the path from S to D , which it can compute using the topology table.

duration to equal the duration field in the S-CTS. Note that because the S-CTS takes some time to travel from the SEN, and the duration field is relative to the SEN, the NAV duration slightly overestimates the handshake duration. We ignore this minor inefficiency for this version of the design.

- 9) The SSN transmits the DATA which is relayed in the pipelined mode. Each relay uses the transmit and receive frequencies that were set when the S-CTS went back. The DATA contains as the "destination" the SEN and when it reaches the SEN a CRC is performed. If there are more packets in the current burst, the "more fragments" flag in the Frame Control field is set. Sentinels use this information to increment their BUSY durations.
- 10) If the CRC is successful, an S-ACK is relayed to the SSN. Otherwise, the DATA is silently discarded. The S-ACK is relayed in non-pipeline mode. A node, including a sentinel, receiving the S-ACK changes the floor state to NOT-BUSY. It is also changed to NOT-BUSY after the expiration of the duration timer (this is in case the S-ACK is lost).
- 11) The SSN either times out and retransmits the same DATA or sends the next DATA in the burst, if any.
- 12) If the SEN is not equal to the final destination of the packet, it now becomes the SSN. After a random duration after the duration period terminates, it initiates another segment toward the destination (in other words, we start again from step 2).

Let $F(t_i)$ and $F(r_i)$ denote the frequencies of the transmitter and receiver respectively at node i . We note that by virtue of 7,

$$\begin{aligned} F(t_i) &= F(r_{i+1}), i = 0, 1, 2, \dots, (k-1) \\ F(t_i) &\neq F(r_i), i = 0, 1, 2, \dots, k \\ F(r_i) &\neq F(t_{i+1}), i = 1, 2, \dots, k \end{aligned}$$

Thus, pipelined, full duplex relaying without intra-path interference is achieved conflict-free using just three frequencies.

An example of the PAC concept is illustrated in Figure 6 on a Manhattan grid where each node can

both interfere and communicate with only its four nearest nodes⁸. We assume that the transmission time for each hop for control packets is 1 unit and DATA is 10 units. We ignore propagation and all other delays. Suppose there are no flows to begin with. Then, flow 1 is initiated from S_1 to D_1 . The frequencies are set up as shown in the figure (see caption for legend information). Notice that the frequencies are assigned in round robin order starting from the *destination*, not the source, because it is the S-CTS that carries and is used to assign frequencies. This sets up the sentinel nodes (shown darkened) to have a floor state of BUSY with a duration value of 18 units. Then the DATA starts.

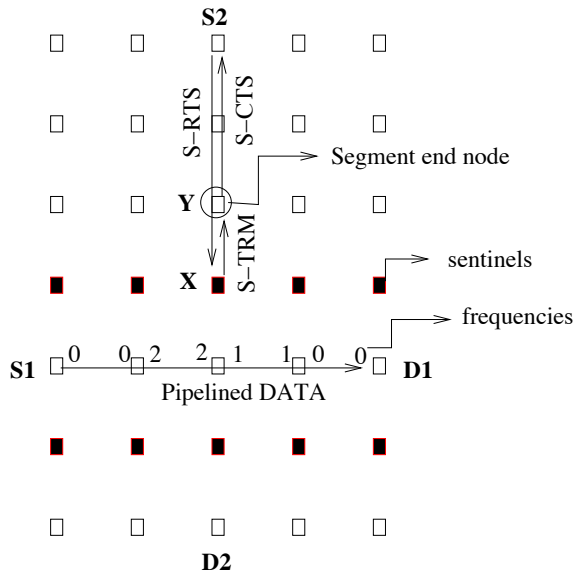


Fig. 6. Example illustrating segment termination and sentinel nodes

While this is ongoing, flow 2 is initiated from S_2 to D_2 . When the S-RTS for this reaches the the sentinel for flow 1, namely node X which has a floor state of BUSY, it responds with a S-TRM (segment terminate) frame. The node Y preceding node X in the path then becomes the segment-end-node, and sends an S-CTS back to S_2 .

S_2 sends the DATA packet (in pipelined mode) which at Y is sent up to PAC. Provided the DATA is received error-free, Y sends an ACK to S_2 . When flow 1 ends, (S-TRM will contain the duration from node X), flow 2 is re-initiated with Y as segment-start-node. A backoff algorithm is used to

ensure that multiple flows waiting for flow 1 to finish do not all start at the same time. Note that the segment needs to terminate not at the sentinel itself but before it because the sentinel’s DATA frequencies may be busy from flow 1 packets. Note also that the S-RTS and S-TRM do not interfere with those DATA packets being on a completely different frequency.

All error detection and correction is done only at the end of each segment. Thus, nodes within a segment may relay a packet with (accumulated) errors. The number of errors will depend upon the segment length and the nature of channel coding, if any, available in the TX/RX chain used. We note that channel coding allows the receive chain to correct errors at relay nodes without violating cut-through pipelining. If block encoding is used, this functionality may also be done as part of transit control, still retaining the ability to pipeline (in the granularity of blocks). We also note that pipelining allows the luxury of choosing short hops with good SNRs and this may significantly reduce the error rate so that even with multiple hops the error rate may be similar to that of a single conventional hop. The segment-end-node sends an ACK if the packet is received without errors, and silently discards the packet if there are uncorrectable errors. The segment-start-node will then retransmit the packet. Sophisticated schemes are possible that increase or decrease the maximum segment length based upon the receipt or non-receipt of ACKS (in a manner reminiscent of “backoff” schemes). These are the subject of future research.

We now compare, using simple numerical analysis, the latencies for conventional (current) systems and S-PAC as a function of the number of hops. We assume a line network, and thus there is one segment per path. For the conventional numbers we assume that the system has multiple channels so that intra-path interference does not exist. This allows a fair comparison – otherwise, the S-PAC which uses multiple channels has an obvious advantage. We ignore the frame handling and turnaround times in both cases but note that this is higher for conventional relaying. Latency is defined as the time between a bit (any bit) leaving the source and

⁸We recognize that communication range is in general shorter than interference range, but since the example is for illustration purposes only, we overlook this for simplicity

arriving at the destination⁹

Let r be the data rate in bits/usec, b be the burst size (number of back-to-back data packets).

The end-to-end latency for conventional (802.11 based) relaying is given by

$$\lambda_{conv} = \frac{h}{r} \cdot (S_{rts} + S_{cts} + b \cdot (S_{data} + S_{ack})) \quad (4)$$

where S_{rts} , S_{cts} , S_{data} and S_{ack} are the length in bits for RTS, CTS, DATA and ACK respectively, inclusive of the PLCP (physical layer) header. The PLCP header is 192 bits, the RTS, CTS and ACK frames alone (without the PLCP) are 160 bits, 112 bits, and 112 bits respectively [6]. We assume a 400 byte payload exclusive of all headers¹⁰ The RTP/UDP/IP headers amount to 40 bytes, and the 802.11 MAC only header is 34 bytes.

Thus, we have the total sizes in bits as

$$S_{rts} = 352, S_{cts} = 304, S_{ack} = 304, S_{data} = 3984 \quad (5)$$

From 5 and 4, the latency for conventional networks is

$$\lambda_{conv} = \frac{h}{r} \cdot (656 + b \cdot 4288) \text{usec} \quad (6)$$

We now consider end-to-end latency of Simple PAC with ROPL. In this case, an S-RTS, S-CTS and S-ACK travels h hops each in non-pipelined mode. Note that this is irrespective of the number of segments, that is, if there are multiple segments, there are multiple different S-RTS/S-CTS/S-ACK packets that together travel h hops each. The data packet, however, is in pipelined mode except at segment-ending nodes. At other nodes, it only incurs the time to receive and process the ROPL header.

Let S_{Srts} , S_{Scts} , S_{Sdata} and S_{Sack} be the per-hop transmission times for the S-RTS, S-CTS, S-DATA and S-ACK respectively, and S_{ropl} is the size of the ROPL header, and N_{sen} is the number of segment end nodes. Let L_s be the maximum segment length.

⁹An alternative would be to define it as the time between (the first bit of) a packet leaving the source and (the last bit of) the packet arriving at the destination, in which case an additional packet transmission time should be added to the account. The difference is immaterial for comparison purposes since we use the same metric for both conventional and S-PAC

¹⁰Assuming a real-time voice application going over RTP/UDP/IP/802.11, the total header size is 98 bytes. The payload size is extremely flexible for most applications, but to have an efficiency of at least 25%, we need the payload to be about 400 bytes, for a total of about 500 bytes per packet.

For simplicity, let us assume that L_s is a factor of h . Then, the latency for S-PAC is given by

$$\begin{aligned} \lambda_{pac} &= h \cdot \frac{S_{Srts}}{r} + h \cdot \frac{S_{Scts}}{r} + \\ & b \cdot h \cdot \frac{S_{Sack}}{r} + b \cdot (h - N_{sen}) \cdot \frac{S_{ropl}}{r} + \\ & N_{sen} \cdot b \cdot \frac{S_{Sdata}}{r} + N_{sen} \left(\frac{S_{Strm}}{r} + \frac{S_{Srts}}{r} \right) \\ &= \frac{h}{r} \cdot (S_{Srts} + S_{Scts} + b \cdot S_{Sack} + b \cdot S_{ropl} + \\ & \frac{N_{sen} \cdot b}{h} (S_{Sdata} - S_{ropl}) + \frac{N_{sen}}{h} (S_{Strm} + S_{Scts})) \\ &= \frac{h}{r} (S_{Srts} + S_{Scts} + b \cdot (S_{Sack} + S_{ropl}) + \\ & \frac{N_{sen}}{h} \cdot (S_{Strm} + S_{Srts} + b \cdot (S_{Sdata} - S_{ropl}))) \end{aligned}$$

Note that because of pipelining, each relay node (non SENs) only incurs a cost of S_{ropl}/r and not S_{Sdata}/r . Also, for each SEN, there is an additional S-TRM and S-RTS transmission that needs to be counted.

The ROPL header size is 290 bits, and the lengths of the S-RTS, S-CTS, S-DATA, S-ACK, and S-TRM from Figure 5 (without ROPL header) are 176 bits, 112 bits, 96 bits, and 144 bits respectively. The S-DATA packet is 400 bytes plus the 40 byte RTP/UDP/IP headers plus 96 bits in the PAC header.

Thus, the sizes in bits are

$$\begin{aligned} S_{Srts} &= 466, & S_{Scts} &= 402, & S_{Sack} &= 386, \\ S_{Strm} &= 434, & S_{Sdata} &= 3906 \end{aligned}$$

The number of segments N_{sen} is $\lceil (h/L_s) - 1 \rceil$. Substituting for N_{sen} and using size values from above, we have

$$\lambda_{pac} \leq \frac{h}{r} \cdot (868 + b \cdot 676 + \lceil \frac{h}{L_s} - 1 \rceil \cdot (\frac{900 + b \cdot 3616}{h})) \quad (7)$$

The latency reduction factor for PAC over conventional is thus

$$\Gamma_{pac}^{conv} = \frac{\lambda_{conv}}{\lambda_{pac}} = \frac{656 + b \cdot 4288}{868 + b \cdot 676 + \lceil \frac{h}{L_s} - 1 \rceil \cdot (\frac{900 + b \cdot 3616}{h})} \quad (8)$$

Figure 7 shows a plot of latency gain when the segment length is 10 for various values of b (h assumed a multiple of 10) using equation 8.

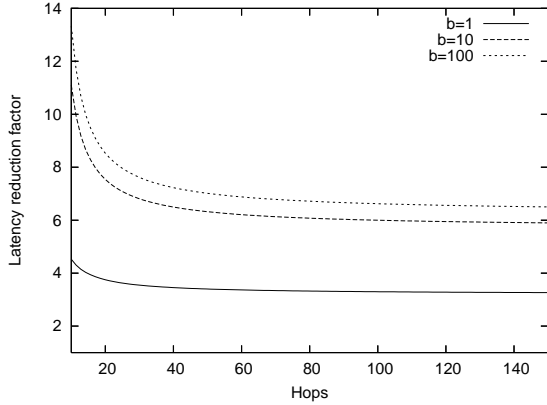


Fig. 7. Latency reduction factor of Simple PAC over conventional forwarding for different packet burst lengths.

The latency reduction factor is nearly constant for higher h 's. Noting that $\lceil (h/L_s) - 1 \rceil \leq h/L_s$, we can get a conservative (lower bound) estimate of the latency reduction factor that is independent of h , as

$$\Gamma_{pac}^{conv} = \frac{\lambda_{conv}}{\lambda_{pac}} \geq \frac{656 + b \cdot 4288}{868 + b \cdot 676 + \frac{900 + b \cdot 3616}{L_s}} \quad (9)$$

Figure 8 shows a plot against L_s for various values of b . As expected, S-PAC does better when the segment length is longer since a packet can be cut-through switched for a longer time before needing to be fully received. At a segment length of 1, the scheme falls back to the conventional scheme (notably, there is no degradation in performance). At reasonable L_s , say 15, there is a 2.5x to 5x reduction. Note that this analysis is under a single stream so the main advantage of PAC – not having to repeat MAC contention, is not utilized. Given this, it is remarkable that even a very simple protocol like S-PAC achieves significant gains.

One negative feature of Simple PAC is that there may be a significant lag between acquiring the floor and using it. In particular, consider a relay node R within a single segment path. When it transmits the multi-hop S-RTS, it acquires the floor (blocks its neighbors from transmitting). However, for R to actually *use* the acquired floor, that is, transmit DATA, it takes *two path roundtrip* times during which the neighbors are *unnecessarily* blocked. In

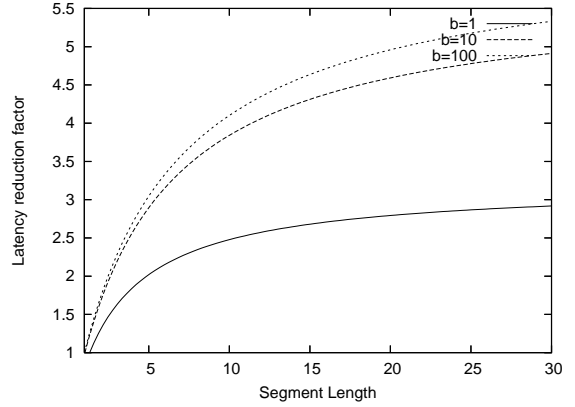


Fig. 8. Latency reduction factor of Simple PAC over conventional forwarding as a function of segment length.

contrast, in conventional transport, only one hop is reserved at a time and so the wastage is only on the order of the hop roundrip time.

While this does not affect latency performance on a relative basis, the question arises whether there is an impact on relative capacity. To study this comprehensively requires analysis beyond the scope of this paper. Here we present a rough analysis with a view to obtaining insight into the tradeoffs.

To simplify the analysis, we have invented and used the notion of *spatial disuse*. The spatial disuse ψ_f created by a flow f is the total amount of time that nodes have been blocked from transmitting or receiving packets other than those belonging to flow f . While spatial disuse does not measure capacity or its wastage directly, it is strongly inversely correlated. In particular, the relative spatial disuse between two schemes can be taken as an inverse measure of relative capacity with reasonable confidence.

We consider a Manhattan grid as in section III-B, with each node being in communication and interference range of eight of its neighbors. A single segment path goes from source S to destination D as shown in Figure XXX. All of the nodes (and only the nodes) in the grid that are affected by this path are shown.

First consider conventional transport using 802.11. Let T be the total time for the handshake (per hop) including RTS, CTS, DATA and ACK. Shown in Figure 9 alongside each node is the time that that node is blocked. For example, consider node M . Nodes A , B and C are its path neighbors.

Node M is blocked when A is the receiver of a frame, when A sends the frame again (and B receives it), when B sends it (and C receives it) and when C sends it, making a total of 4 handshake times, or $4 \cdot T$. The blocking times for nodes on the boundary (one hop from the source or destination) are less and are as shown in the figure. The spatial disuse for conventional is given by

$$\psi_{conv} = (3 \cdot (4 \cdot (h - 3) + 2 \cdot 3 + 2 \cdot 1)) \cdot T_H^{conv} \quad (10)$$

where T_H^{conv} is the time for a handshake of RTS, CTS, a burst b of DATA-ACK. Denoting the data rate by r , and the sizes as S_{rts} , S_{cts} , S_{data} and S_{ack} , we have

$$\psi_{conv} = \frac{12 \cdot h \cdot (S_{rts} + S_{cts} + b \cdot (S_{data} + S_{ack}))}{r} \quad (11)$$

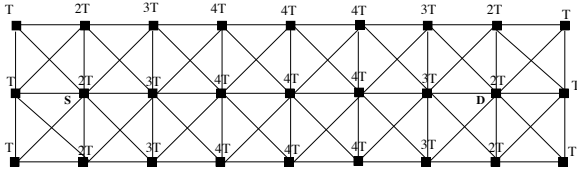


Fig. 9. Amount of time, in units of T , that each node is blocked from sending, when a packet is sent from S to D in conventional systems.

Now consider PAC. Consider a node M that is m hops from the source toward the destination as shown in Figure XXX (b). This node and its neighbors are free until the time the S-RTS from the source reaches M , and busy thereafter. Further, it is busy for a period that is the sum of the time taken for the S-RTS to reach the destination, for the S-CTS to go from the destination to the source, and a burst b of S-DATA and S-ACK to complete. And since there are three such rows, we have

$$\psi_{pac} = 3 \cdot \left(\sum_{m=1}^h (h - m + 1) \frac{S_{rts}}{r} + \frac{(h + 3)}{r} (S_{cts} + b \cdot (S_{data} + S_{ack})) \right)$$

Using size numbers from before, the *disuse ratio* is thus

$$\Delta_{conv}^{pac} = \frac{699 \cdot (h + 1) + (3h + 9) \cdot (290b + 836)}{3624 + 47808b} \quad (12)$$

The disuse ratio is plotted in Figure 10.

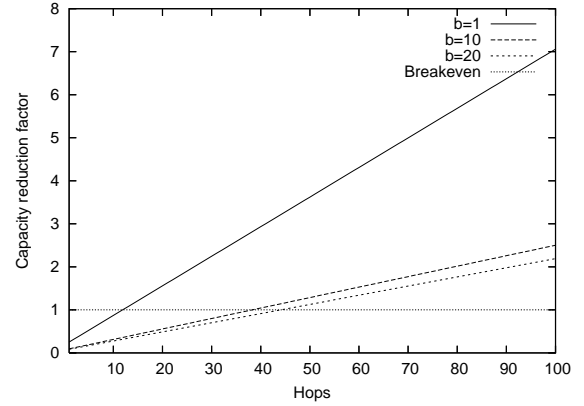


Fig. 10. Disuse ratio (Conventional disuse divided by S-PAC disuse. When disuse ratio is less than one, S-PAC is doing better than Conventional in terms of capacity.

Figure 10 shows that there is a tension between the advantage of quickly getting a packet transmission completed and the disadvantage of waiting around without utilizing the acquired floor. There is a cut-off number of hops beyond which S-PAC has more spatial disuse than conventional. The cut-off point depends upon the burst length b – the more b is, the higher the hop number at which the S-PAC becomes worse. What is interesting is not that S-PAC wastes capacity (this is intuitively obvious) but that for smaller networks, we can actually get the latency gains without losing capacity gains.

Nonetheless, this calls for more intelligent Path Access Control, where the DATA packet might follow the S-RTS rather than wait for the S-CTS. This idea, termed *tailgating PAC* is under research.

IV. EXPERIMENTAL ANALYSIS

We have implemented a simulation model for the ultra low latency system described in the previous section using NS-2 (version 2.28) [14]. We chose NS-2 due to its relative ease of use. NS-2 is a discrete event simulator whose source code is split between C++ for core implementation and OTcl for configuration and scripting. NS-2 supports wireless networks such as MANETs and Wireless LANs and its implementation includes an 802.11 module.

The model includes the Simple Path Access Control (S-PAC) and the Relay Oriented Physical Layer (ROPL). To implement these, we have modified the C++ implementation of 802.11 (ns-2.28/mac/mac-802_11.{h,cc}) In particular we have implemented

two major changes to the source code. The first change is for S-PAC and enables multihop support of 802.11 MAC frames. That is, we augmented the RTS,CTS,DATA,ACK frames with a target destination field. On the transmission path of the 802.11 MAC, we inserted appropriate hooks to allow the update of relevant frame fields, and to allow access to the forwarding table at each hop. On the reception path, intermediate hops can relay any 802.11 MAC frame while only segment termination nodes can generate CTS and ACK frames.

The second change is for the ROPL in which pipelining of frame bits occurs. In our model, when the first bit of frame is detected by the WirelessPHY module, the entire frame is retrieved and forwarded after a delay equivalent to the time to transmit an RTS-sized block. A two-orthogonal-channel model (one channel for control traffic and another for data) has been added to support ROPL. All frames are transmitted based on information contained in the *transit table*. The transit table is an array (indexed by source and destination) of *keep/relay/discard* enumerations and is populated based on information learnt through 802.11 frames and/or the forwarding table. For carrier sensing to work effectively in our PAC-enabled environment, the NAV duration has been set to a function of the number of hops and transmit times of RTS and DATA frames.

With the aforementioned NS-2 changes, an instance of a multihop wireless mesh network is created by placing a set of N^2 nodes in a $N \times N$ square area. Nodes are stationary and are placed in a Manhattan grid. Routes among nodes are statically configured with the help of a *NO Ad-Hoc Routing Agent* (NOAH) [15]. In all our experiments, we assume a channel rate of 11Mbps for both 802.11 and S-PAC. We have experimented with two modes of PHY operation using S-PAC. The first mode is ROPL while the second mode is the conventional 802.11 PHY in which the entire packet is received by the PHY, passed to upper layers (including Routing) before possible re-contention on the same channel. For traffic generation, a fraction p of all nodes are selected randomly to be sources of traffic while nodes, not too close from sources, are picked randomly as destinations. Each source can generate CBR traffic at a rate of R packets per second. To speed up simulations, we use longer packets (of

8000 bits in base size) instead of many packets. This abstractly models back-to-back packet transmissions within RTS/CTS round trips. Each simulation lasts S seconds. A traffic session starts at a time uniformly distributed between 1 and $(S - 2D)$ seconds and lasts for a duration uniformly distributed around D seconds (namely $[0.8D, 1.2D]$).

A. PAC Performance

We now analyze the performance of PAC versus that of 802.11 as a function of the network size and the offered load. Both aforementioned PHY modes are considered for PAC. In all our experiments, $(R, D, S) = (1, 50, 200)$ ¹¹. Metrics of particular interest are

- *Latency*: The end-to-end delay of the application traffic.
- *Average number of segments*: The total number of segments per path over total number of paths.

In all cases considered, the *packet delivery rate* is 100 %.

Figure (11) depicts the latency as a function of the number of hops for a single traffic flow (no contention). In this experiment, 802.11 has been considered with and without use of RTS/CTS. Not surprisingly, PAC outperforms both 802.11 cases by at least a factor of 4. This result confirms our numerical analysis.

For the remaining section, we shall consider that RTS,CTS frames are always in use with 802.11. Figure (12) shows a set of results with multiple traffic flows ($p = 12\%$). In all associated figures, the latency is depicted as a function of the network size for various rates of flow and packet burst. PAC still outperforms 802.11 but the gap increases significantly with an increased traffic burst. Overall we obtain 3.5x to 8x gain (depending upon burstiness) at moderate, and realistic loads. Gains are higher with smaller number of higher-rate flows than with a larger number of lower-rate flows. analysis.

Figure (13) shows results with PAC operating on top of a conventional PHY. Unlike earlier figures, the X axis has offered load rather than number of hops because of the difficulty in controlling the number of hops per flow while preserving randomness. Without ROPL, PAC does slightly worse than

¹¹Values of these parameters were chosen mainly for their simplicity.

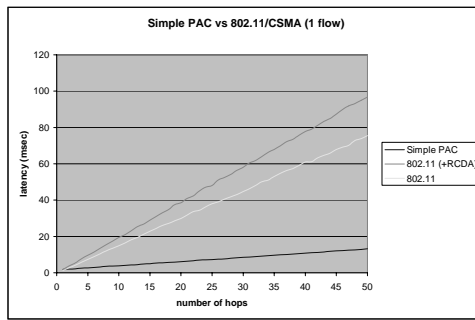


Fig. 11. Single flow (one source to one destination), relay-oriented PHY.

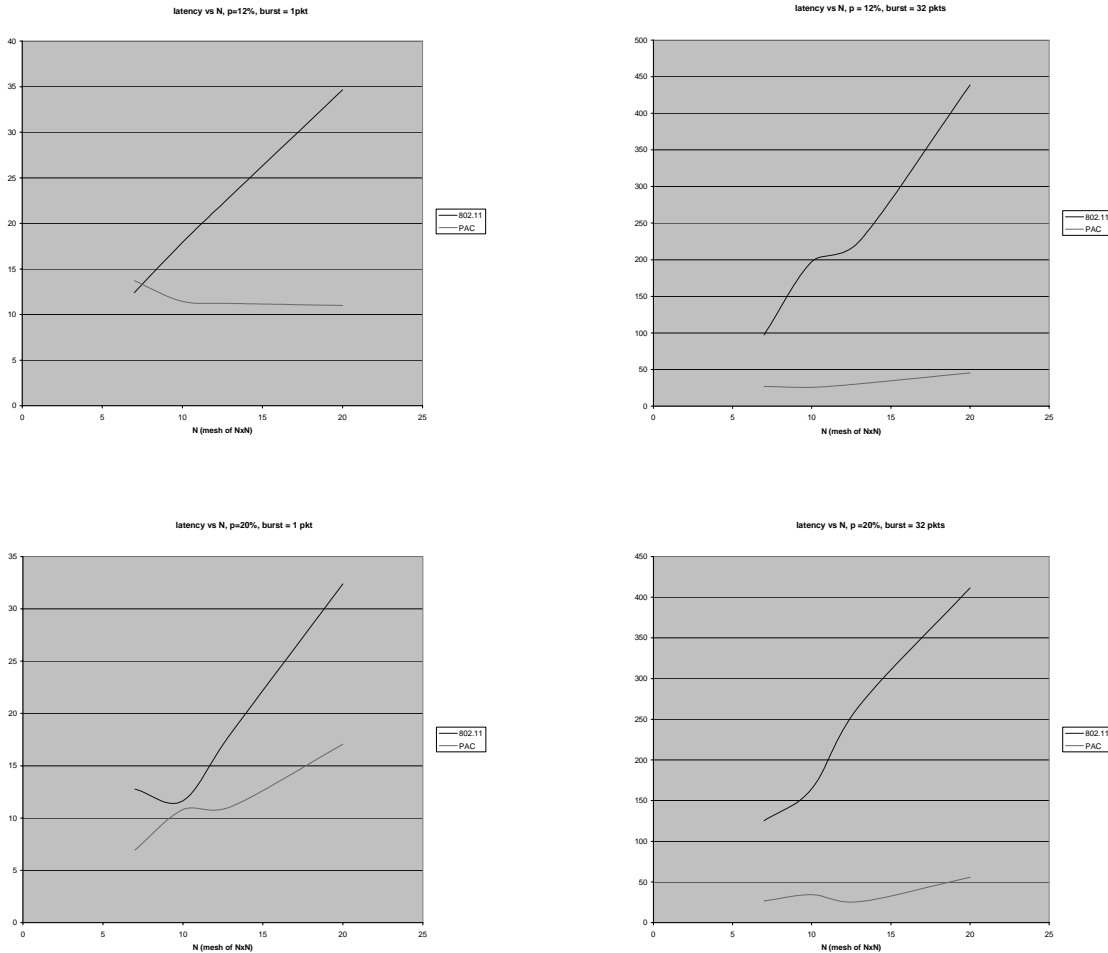


Fig. 12. Multiple flows (many source-destination pairs) with various bursts, relay-oriented PHY.

802.11 in terms of latency. Indeed all PAC frames now have to re-contend for channel access at each hop, which increases the per-hop delay. This result shows that PAC must be used in conjunction with ROPL to obtain maximal gains.

V. IMPLEMENTATION CONSIDERATIONS

One question on the Relay Oriented Physical Layer is: how hard is it to implement? Unlike most other MANET solutions which work at the MAC layer and above, our proposal delves into the physical layer, requiring changes to what is considered to be the radio “hardware”.

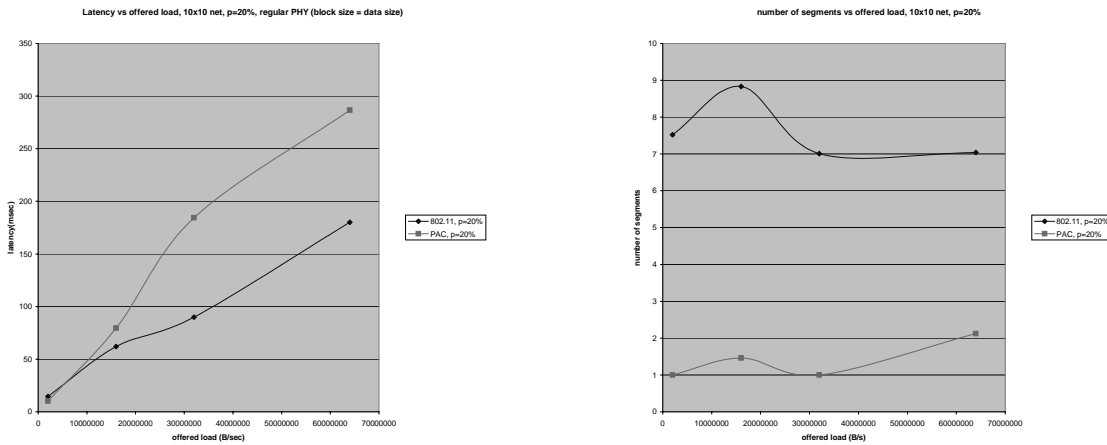


Fig. 13. Multiple flows (many source-destination pairs), conventional PHY.

In this section, we show why this is not as hard as it may appear. First, as noted earlier, the ROPL does not involve any changes in the signal processing elements in the receive or transmit chain. Given any legacy physical layer RF and modem components, ROPL simply adds processing at the end of the receive chain and at the head of the transmit chain. Thus, if there is an on-board CPU, simple code can be written to implement the algorithm described in section III-A. Alternatively, one can use a Field Programmable Gate Array (FPGA) into which the logic is burnt in. The floor state and the relay table are loaded into the FPGA as and when they change. If the transmit and receive processing (modem functions) are part of an existing FPGA whose source code (e.g. Verilog) is available, they can be upgraded with the same logic. Thus it is relatively easy for a chipset manufacturer (such as Atheros), or even a wireless LAN card vendor (such as Cisco) to build an ROPL.

A more exciting and flexible avenue is through the use of a software radio. A software radio gets code as close to the antenna as possible [17]. In contrast to conventional radios which processing is done using analog circuitry and digital chips, in a software radio the software defines the transmitted waveforms and demodulates the received waveforms. Software radios are beginning to find increasing usage both in the military and commercial [16] worlds.

We consider a design based on the open source GNU Software Radio [17]. While this is currently

not fully capable of full-fledged packet based communications and protocols, work is underway that will make it a viable platform for hosting MANET-like protocols. Specifically, the DARPA ACERT program is investing heavily in the GNU Radio to make it usable by researchers. It will have a full digital transceiver and a flexible MAC protocol. Given these, to upgrade to an ROPL is simply a question of additional code.

The Gnu Radio architecture is based upon a number of processing blocks written in C++ that are strung together in real-time using the Python language. The processing blocks perform such receive functions as demodulation, decoding, descrambling etc. and transmit functions such as modulation, coding, scrambling etc. When a block B_1 is connected to block B_2 , the bit stream that is out put from B_1 becomes the input stream for B_2 on which it acts. While the Gnu Radio, being entirely software, can run on any basic hardware (note that Gnu Software Radio does not completely eliminate hardware because you still need circuitry to bring a signal down to the digital domain), a specific hardware solution called the Universal Software Radio Peripheral (USRP) is available for purchase [18]. The USRP consists of a small motherboard containing up to four 12-bit 64M sample/sec ADCs, four 14-bit, 128M sample/sec DACs, a million gate-field programmable gate array (FPGA) and a programmable USB 2.0 controller. Each fully populated USRP motherboard supports four daughterboards, two for receive and two for transmit. RF front ends are

implemented on the daughterboards.

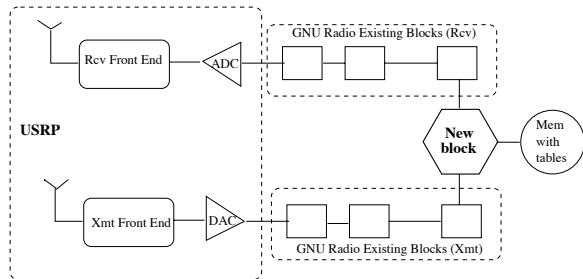


Fig. 14. Implementation outline of ROPL using GNU Radio

The implementation sketch for ROPL using the GNU Radio is given in Figure 14. The solution consists of creating a new processing block (called *relay_processing*, and having the output of the last processing block in the receive chain be connected to the input of *relay_processing* and the output of *relay_processing* be connected to the first block in the transmit chain. The *relay_processing* block reads the header, looks up the routing tables in the memory, decides whether or not to relay and activates the sending of the stream, as was discussed in more detail in section III-A. Writing a new block is relatively easy by inheriting from *gr_block* class, and is described in [19].

The PAC is, of course, not so hard to implement even in conventional hardware, but is even more easy in Gnu Radio environment. Under the aforementioned DARPA ACERT project, common elements needed for MAC are being built and can easily be adapted for PAC.

VI. CONCLUDING REMARKS

If current trends in MANET topologies and real-time applications continue, the ability of conventional MANET mechanisms to support the end-to-end latency requirements of applications will be under serious threat. Incremental advances in MAC and network layer will be insufficient to provide the dramatic reduction in latency that is required.

We have described two key components of a solution for ultra low latency packet transport in MANETs. Our Relay-Oriented Physical Layer (ROPL) performs cut-through switching using orthogonal frequencies for transmit and receive, and by pipelining the received bits as soon as the first few bytes of the packet are received and a decision

is made on relaying. A proactive routing protocol injects next-hop forwarding table information into the physical layer. The ROPL cuts the switching time by a factor of more than 10x at expected data rates and packet size.

We also described S-PAC – a simple mechanism for path access control. S-PAC extends the hop-oriented RTS/CTS/DATA/ACK handshake to a path oriented regime. It works in units of *segments*, multiple segments constituting an end-to-end path. The S-PAC terminates and re-initiates segments, allows packet bursts, assigns frequencies and creates *sentinel* nodes for protecting the transfer. Simple numerical analysis shows that the S-PAC reduces the end-to-end latency by more than a factor of 5, and may even increase the capacity upto a certain number of hops.

Using NS-2, we have conducted experiments on an abstract model of the ROPL and S-PAC on a Manhattan grid topology with random streams of traffic. Our simulations show a gain of 3.5x to 8x in latency over the baseline.

A number of exciting research directions need to be explored. The S-PAC, while simple, has a number of obvious inefficiencies that can be removed. New path access control mechanisms where the DATA packet does not have to wait at the source till the RTS/CTS is complete also need to be investigated. TDMA-oriented PAC is another promising but unexplored direction. The concept of adaptively controlling the segment length based on error and other characteristics is a challenging yet fruitful topic. Finally, implementing the ROPL and the PAC into a prototype testbed, either using conventional hardware or the GNU Radio will be the ultimate validation of this concept.

REFERENCES

- [1] <http://www.darpa.mil/ato/solicit/WANN/index.htm>
- [2] J. M. Kahn and R. H. Katz and K. S. J. Pister, "Next century challenges: mobile networking for smart dust", *Proc. ACM MobiCom*, 1999.
- [3] J. Bicket, D. Aguayo, S. Biswas, R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network", *Proc. ACM Mobicom*, 2005.
- [4] R. Ramanathan, "Challenges: A Radically New Architecture for Next Generation Mobile Ad Hoc Networks," *Proc. ACM Mobicom*, 2005.
- [5] <http://www.darpa.mil/ipto/Programs/acert/index.htm>
- [6] M.S. Gast, e Definitive Guide", O'Reilly and Associates, April 2002.
- [7] P. Kermani and L. Kleinrock, "Virtual Cut-through: A New Computer Communication Switching Technique", *Computer Networks*, 3(3):267–286, September 1979.
- [8] E. Leonardi, F. Neri, M. Gerla, P. Palnati, "Congestion Control in Asynchronous High-Speed Wormhole Routing Networks", *IEEE Communications Magazine*, Nov. 1996.

- [9] A. Acharya, A. Misra, and S. Bansal, "A Label-switching Packet Forwarding Architecture for Multi-hop Wireless LANs," *IBM Tech. Rep.*, 2002.
- [10] D. Raguin, M. Kubisch, H. Karl, A. Wolisz, "Queue-Driven Cut-through Medium Access in Wireless Ad Hoc Networks" *IEEE Wireless Communications and Networking Conference (WCNC)*, Atlanta, Georgia, USA, March 2000
- [11] P. Jacquet, P. Muhlethaler, and A. Quayyum, "Optimized link state routing protocol", IETF MANET Working Group Internet-Draft, Work in Progress
- [12] C. Santivanez, S. Ramanathan, I. Stavrakakis, "Making Link State Routing Scale for Ad Hoc Networks," in *Proc. ACM MobiHoc* Long Beach, CA, Oct. 2001.
- [13] B. Sadeghi, V. Kanodia, A. Sabharwal, E. Knightly, "OAR: An Opportunistic Auto-rate Media Access Protocol for Ad Hoc Networks" *Wireless Networks*, Vol 11, Nos 1-2, Jan 2005
- [14] The Network Simulator, NS-2. <http://www.isi.edu/nsnam/ns>.
- [15] NO AdHoc Routing Agent (NOAH).
<http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/>
- [16] Vanu Inc. <http://www.vanu.com>
- [17] Gnu Software Radio <http://www.gnu.org/software/gnuradio>
- [18] <http://comsec.com/wiki?UniversalSoftwareRadioPeripheral>
- [19] <http://www.gnu.org/software/gnuradio/doc/howto-write-a-block.html>