

FreeMAC: Implementing a Multi-Channel TDMA MAC on 802.11 Hardware

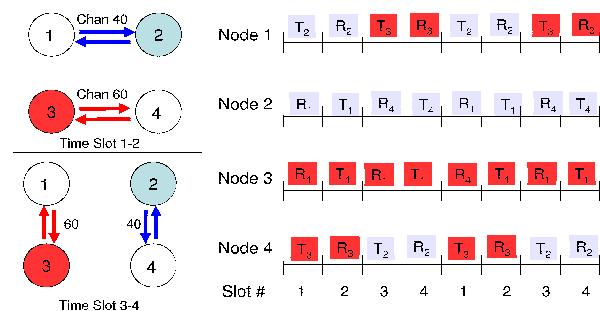
Ashish Sharma, Elizabeth M. Belding
Department of Computer Science
University of California, Santa Barbara CA 93106
{asharma, ebelding}@cs.ucsb.edu

Modern wireless devices offer increased software control over radio communication parameters. In the future, the trend of making the hardware more programmable is expected to grow due to the emergence of cognitive and software defined radio networking. Since a large portion of the MAC protocol is implemented in software, with the firmware providing a set of functional primitives, it is possible to design and implement alternate MAC protocols in real testbeds equipped with commodity 802.11 devices [1, 2, 3]. Validation of protocols on real testbeds helps characterize the additional system constraints that are difficult to capture in a simulated environment. This work demonstrates FreeMAC – a framework that enables the design and implementation of a general class of *multi-channel* MAC protocols, with strict timing requirements, on a typical Linux system.

Multi-channel MAC protocols aim to improve the capacity of a wireless network by time-multiplexing the operation of nodes on orthogonal channels. Generally such protocols rely on precise time scheduling of channel switch operations, usually every 20-80 ms to limit latency. TDMA based MAC protocols require that packets are transmitted during designated time slots. However, implementing such MAC protocols on a Linux system using commodity 802.11 hardware involves several systems challenges. The focus of the FreeMAC platform lies on enabling the primitives for the deployment of such MAC protocols on existing systems. FreeMAC allows tight control over several radio parameters using API functions. We also propose a novel approach of programming the *beacon interval* to invoke periodic hardware interrupts that can be used to bound the latency in scheduling of time-sensitive MAC functions using kernel timers.

We use the FreeMAC framework to implement a simple proof of concept multi-channel TDMA based MAC on a testbed of 4 laptops, each running kernel 2.6.15.7 on the Ubuntu Linux distribution. Each laptop is equipped with an AR5212 chipset-based LinkSys 802.11 a/b/g PCMCIA card. We use the OpenHAL port of MadWiFi - an open source driver for Atheros chipset-based commodity 802.11 wireless devices. In our testbed setup, as shown in Figure , we demonstrate the capability of the FreeMAC framework to implement both single-channel (for nodes 2 and 3) as well as

multi-channel (for nodes 1 and 4) TDMA-style MAC protocols. The TDMA schedule of our testbed comprises of four 50ms timeslots (the slot duration is reconfigurable), operating on two orthogonal channels. Each node transmits in two of these slots and receives on the remaining two. We eliminate per-packet acknowledgements and reduce the contention period to a minimum to ensure a TDMA style MAC implementation.



By exporting the programmability available at the radio hardware as API functions, platforms like FreeMAC can pave the way for increased cross layer interactions and efficient network protocol designs. We plan to use the FreeMAC platform to design and validate dynamic TDMA-style multi-channel MAC protocols for deployment in rural mesh networks.

1. REFERENCES

- [1] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. SoftMAC - Flexible Wireless Research Platform. In *HotNets'05*, College Park, Maryland, USA, Nov 2005.
- [2] A. Sharma and E. M. Belding. FreeMAC: Framework for Multi-Channel MAC Development on 802.11 Hardware. In *PRESTO'08*, Seattle, WA, USA, Aug 2008.
- [3] A. Sharma, M. Tiwari, and H. Zheng. MadMAC: Building a Reconfigurable Radio Testbed Using Commodity 802.11 Hardware. In *WSDR '06*, Reston, VA, USA, Sep 2006.