



MadMAC: Building a Reconfigurable Radio Testbed using Commodity 802.11 Hardware

A. Sharma, M. Tiwari, H. Zheng
Dept. of Computer Science
Univ. of California, Santa Barbara



Motivation

- Proliferation of wireless devices under limited resources
- Need adaptive radio devices
 - Reconfigurable radios for physical layer adaptability
 - Reconfigurable MAC protocols for spectrum access and interference avoidance
- Design adaptive devices
 - Completely new devices, CR, SDR
 - Expensive, not commonly available yet
 - Modifying commodity devices
 - Switch between different MAC protocols
 - Hard, since most are 802.11 devices, CSMA driven



Our Contribution

- Build an adaptive device with reconfigurable MAC layer using commodity 802.11 hardware
 - Wide-availability and low-cost advantage
 - Support multi-channel transmissions to simulate spectrum agility
- MadMAC
 - Provide the capability to transmit a packet at controllable time and frame formats
 - Do not trigger CSMA contention and backoff

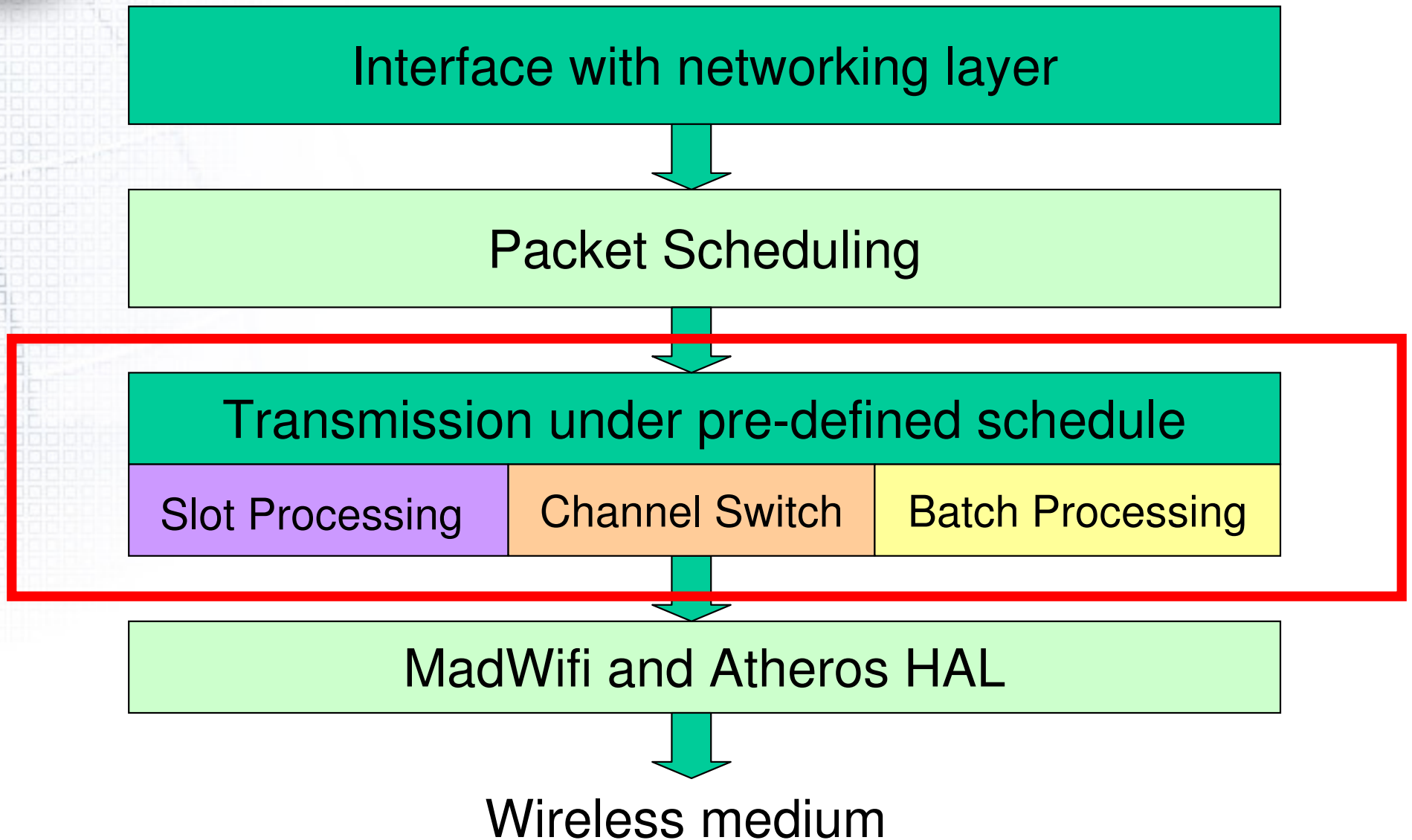


MadMAC

- A kernel-driver built on top of the MadWiFi driver
 - Leverage the open-source programming capability of MadWiFi
 - Interact with radio hardware in real time
- Laptop with Linux kernel 2.6.15-26
- Linksys WPC55AG wireless cards
 - Operate in the monitor mode to disable most CSMA functions
- Focus on implementing a TDMA MAC layer



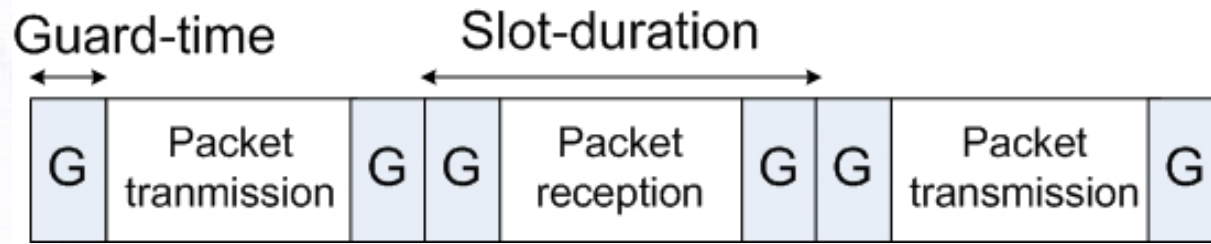
MadMAC Architecture



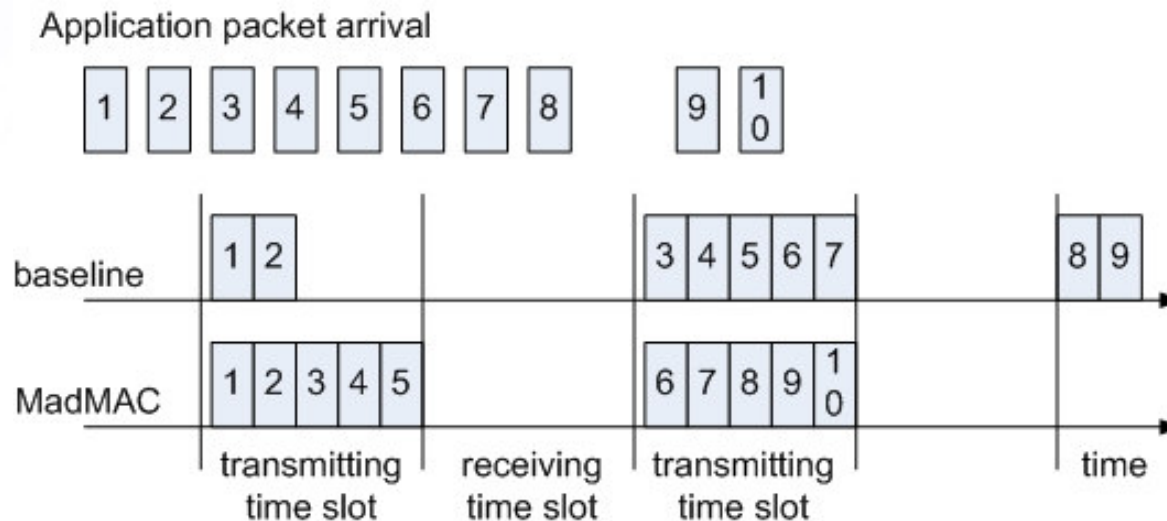


Scheduled Packet Transmission

- Transmit packets at predefined schedules
 - Maintaining a time-slot structure



- Processing multiple packets





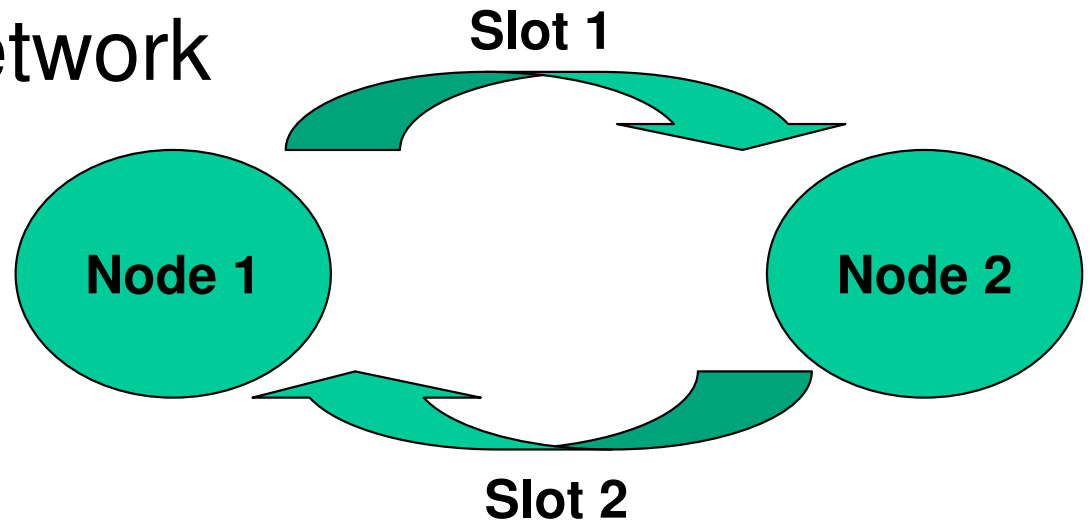
Additional contributions

- Interface with Network layer
 - Convert out-going application packets (*i.e.* IP packets) into MAC frames in a format defined by the MAC protocol; retrieve application packets from in-coming MAC frames.
- Peer synchronization
 - Software based, cluster based synchronization



Experimental Setup

- A simple 2-node network



- Two types of traffic
 - Ping packets
 - UDP traffic (modified Iperf)
- Performance metrics
 - MadMAC processing overhead
 - RTT & Throughput

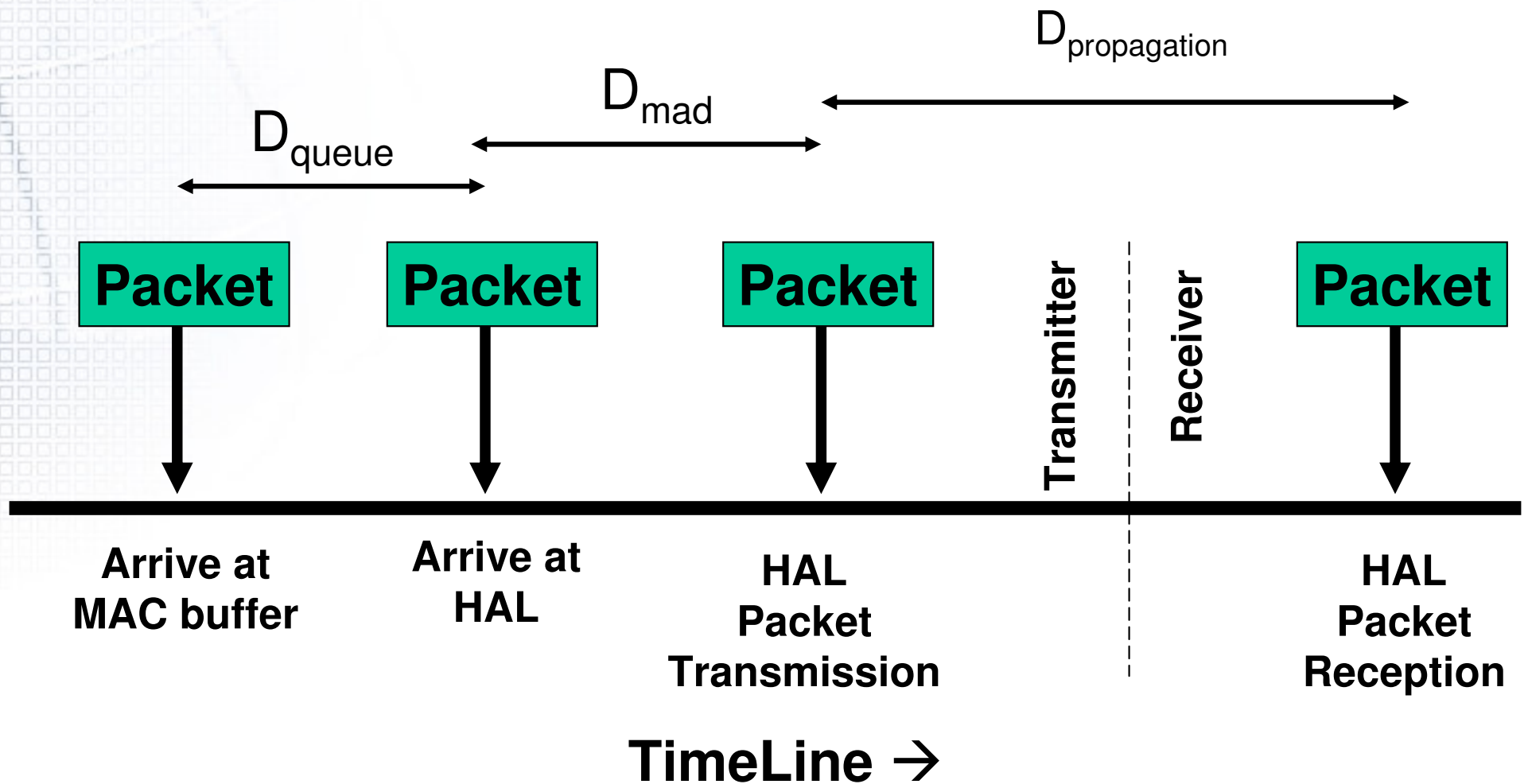


List of Experiments

- MadMAC Processing overhead
- Packet RTT, mean and standard deviation
- UDP throughput (measured using modified Iperf)
- Throughput comparison of different MAC protocols

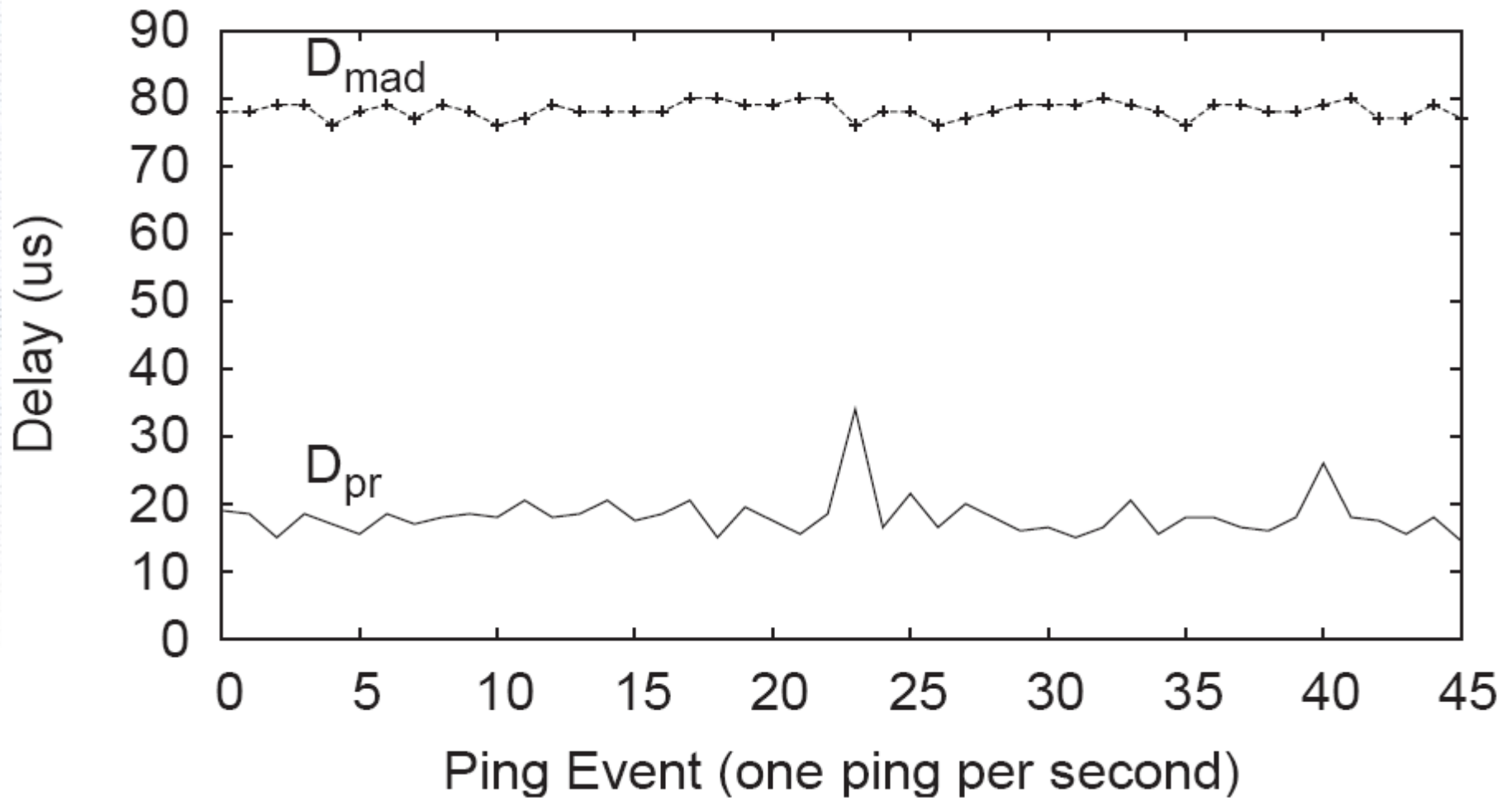


MadMAC processing overhead





Measured Overhead (Ping)



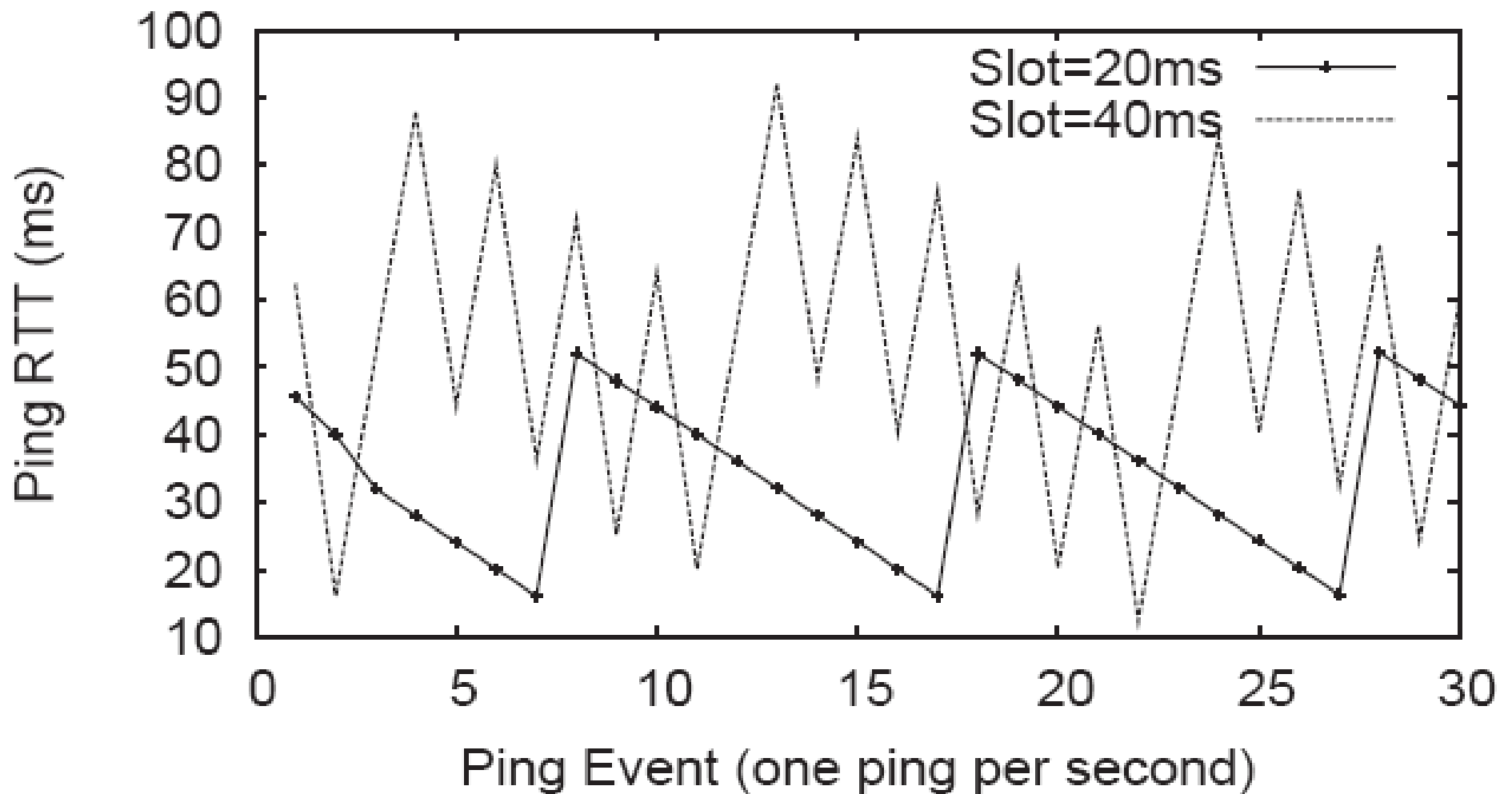


List of Experiments

- MadMAC Processing overhead
- **Packet RTT, mean and standard deviation**
- UDP throughput (measured using modified Iperf)
- Throughput comparison of different MAC protocols

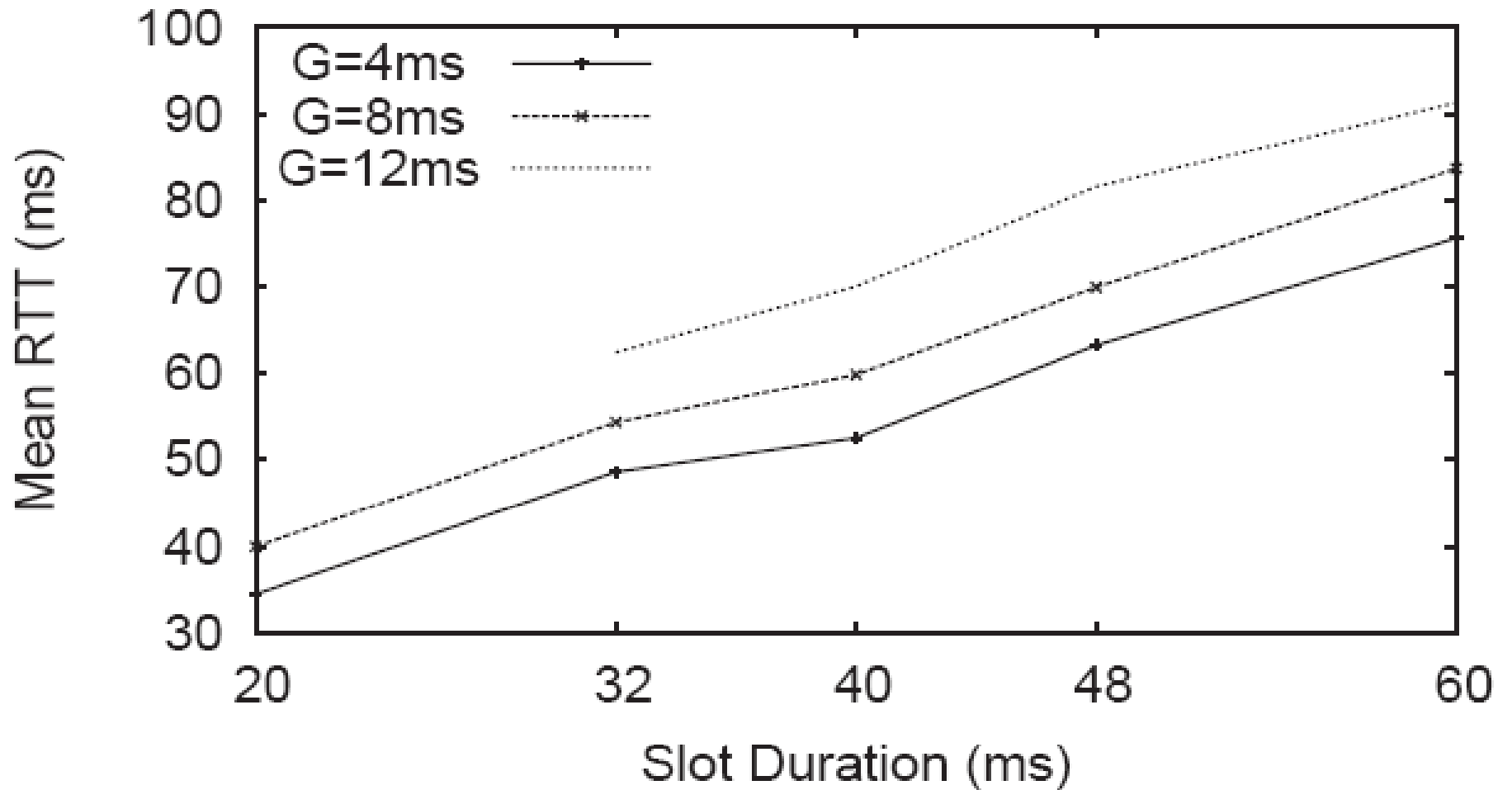


Trace of Ping RTT



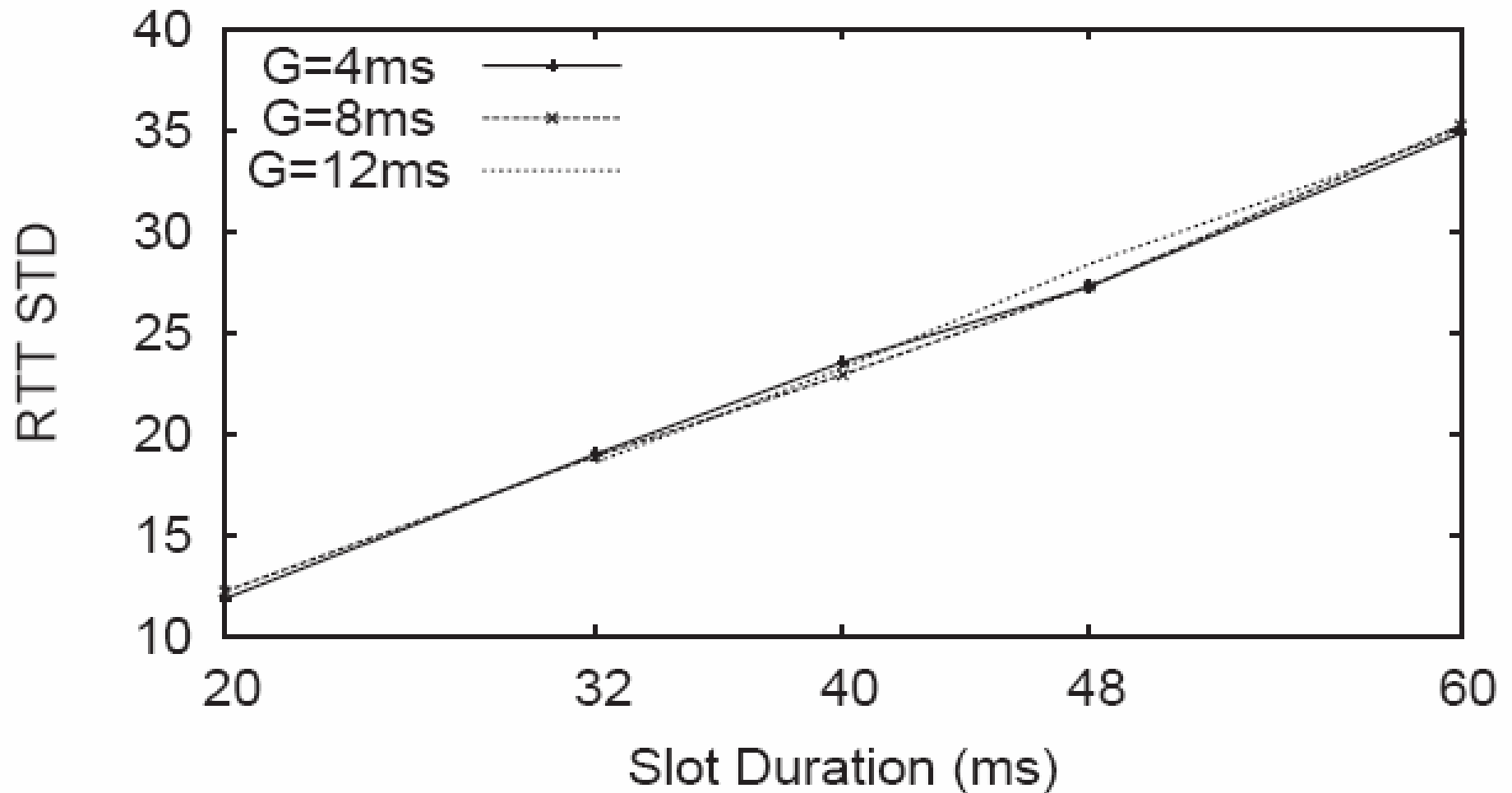


Mean RTT (Ping)





Standard deviation of RTT (Ping)



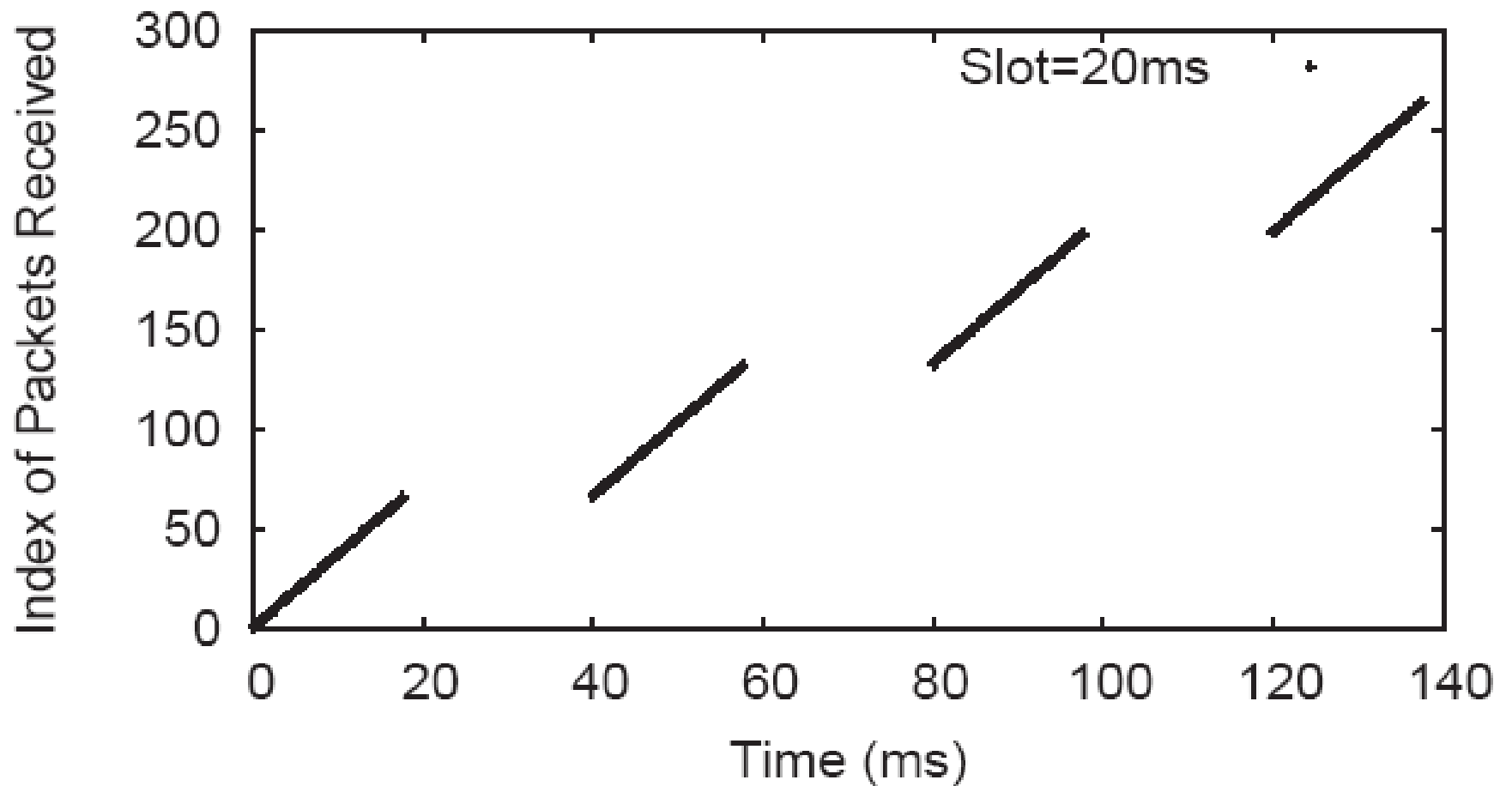


List of Experiments

- MadMAC Processing overhead
- Packet RTT, mean and standard deviation
- **UDP throughput (measured using modified Iperf)**
- Throughput comparison of different MAC protocols

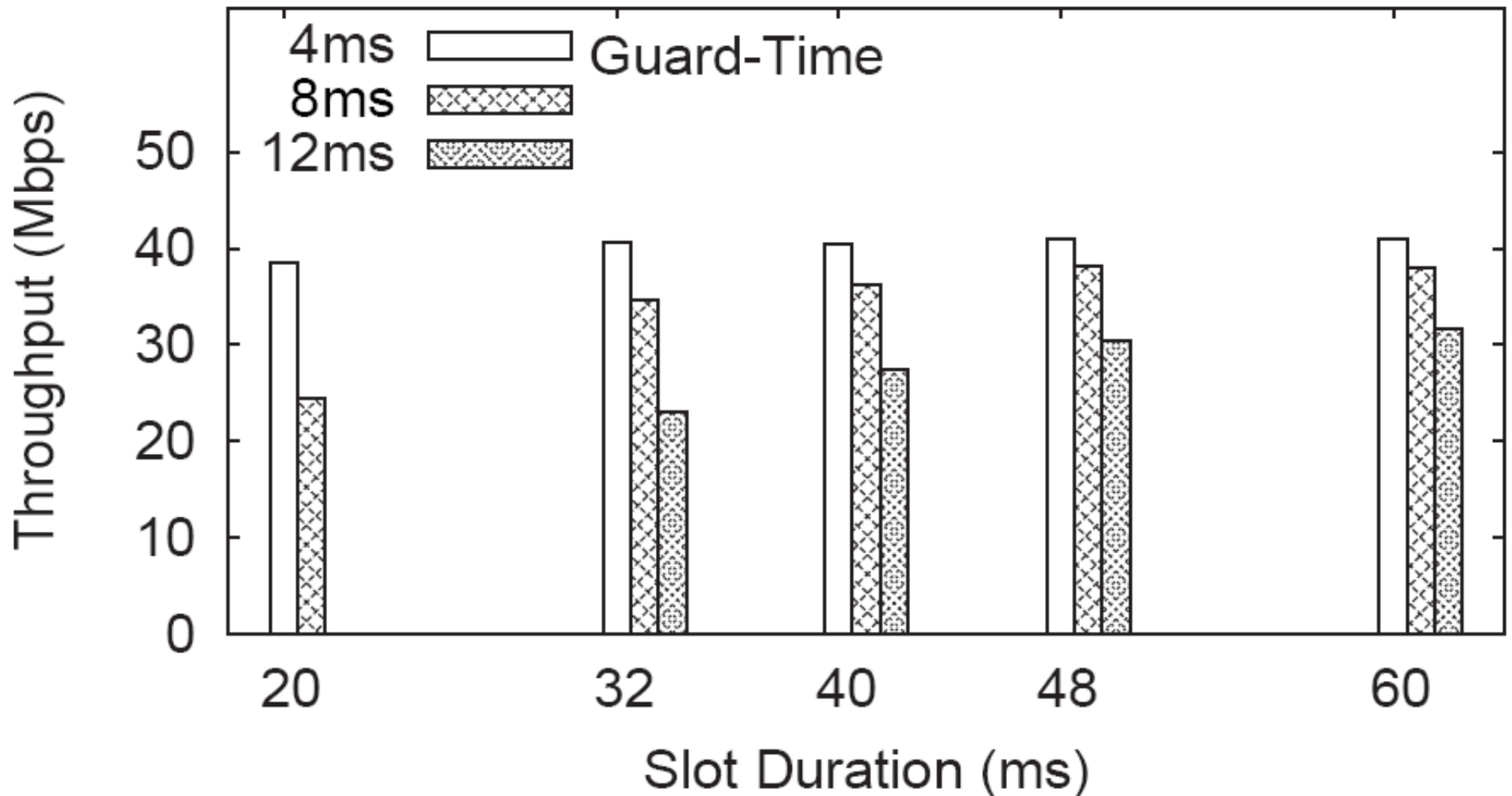


Index of packets received with time (UDP Iperf)





Throughputs for various slot & guard times (UDP Iperf)





List of Experiments

- MadMAC Processing overhead
- Packet RTT, mean and standard deviation
- UDP throughput (measured using modified Iperf)
- **Throughput comparison of different MAC protocols**



Comparative analysis of throughput (UDP Iperf)





Conclusion

- Building a TDMA MAC protocol using commodity 802.11 hardware
- MadMAC: kernel-mode driver on top of MadWiFi
 - Packet transmissions at controllable time and frame formats
- Functional implementation
 - Small processing overhead
 - Verified through 2-node system test
- Future work
 - Expand to a large network
 - Improve synchronization, time scheduling