Energy-Minimizing Idle Listening in WiFi-Equipped Mobile Devices

Kang G. Shin
The University of Michigan

(jointly with Xinyu Zhang)
WiFi for mobile devices

- WiFi: popular means of wireless Internet connection

- WiFi: a main energy consumer in mobile devices
  - 14x more than GSM on cellphones, even in idle mode (w/o packet tx/rx)

Why?
Idle Listening (IL) is expensive!

- WiFi spends most of time in IL
  - Due to the nature of WiFi CSMA
- IL power is comparable to TX/RX
  - Known for WiFi devices
  - Why? All components stay awake and run at full speed during IL
State-of-the-art

- **Sleep scheduling** (802.11 PSM and its variants)

  Sleep scheduling reduces unnecessary waiting (IL) time
  - Client wakes up and performs CSMA *only when needed*

  Is sleep scheduling good enough?
Is WiFi sleep scheduling good enough?

- Analyzed real-world WiFi packet traces
- CDFs of time and energy fractions spent in IL

O1. <10% of time spent for tx/rx
O2. >80% of energy spent in IL for most users
Why?

- **Contention time** (carrier sensing & backoff)
  
  Even if the client knows existence of a packet to send or receive, it must *WAIT* for a *channel access* opportunity.

- **Queuing delay**
  
  Even if the client knows existence of a packet buffered at AP, it must *WAIT* for its *turn* to receive.

Energy/wait cost is shared among all clients.

The more clients, the more wait/energy is wasted in IL for each client!
Key Observations and Idea

• Main observations:
  - IL energy = \( \text{Time} \times \text{Power} \)
  - Rationale: \( \text{Power} \propto \text{Clock-rate} \)

• Key idea:

Existing WiFi
- Constant clock-rate

Our Solutions
- Adaptive clock-rate
Power savings by downclocking

**WiFi**

- IL Power (W)
- Clock rate
- 47.5% saving

**USRP**

- IL Power (W)
- Clock rate
- 36.3% saving
Key challenge: rx packets at low clock-rate

- **Theoretic limit:**
  
  Nyquist-Shannon sampling theorem: to decode a packet, we need
  
  * Receiver’s sampling clock-rate $\geq 2 \times$ signal bandwidth

  Receiver’s sampling clock-rate $\geq$ transmitter clock-rate

- **Challenge:**

  Packets cannot be decoded if the receiver is downclocked
Separate detection from decoding!

Our Solutions

Packet detection is **not** limited by Nyquist-Shannon theorem

* Customize preamble to enable *sampling-rate invariant detection* (SRID)
Two Specific Solutions

- **Energy-Minimizing idle Listening (E-MiLi):** based on self-correlation (MobiCom’11)

- **Gap Sense:** based on energy pulses and gaps (INFOCOM’13)
E-MiLi: Sampling-Rate Invariant Detection (SRID)

How can a packet still be detected even when the receiver operates at low clock-rate?

M-Preamble Design

- M-preamble: $C$ duplicated versions of a random sequence
- Exploit self-correlation between duplicates for pkt detection
- Duplicates remain similar even after down-sampling the preamble
  * Resilient to the changes of sampling rate
Sampling-Rate Invariant Detection, cont’d

M-Preamble Design, cont’d

- **Basic rules:**
  - Self-correlation ≈ Signal’s energy
  - \( \frac{\text{Avg Energy}}{\text{Noise floor}} \) > minimum detectable SNR

- **Enhanced rule:**
  - # of sampling points satisfying basic rule
  \[ \approx \frac{C - 1}{C} \cdot \text{M-preamble length} \]
PHY-layer address filtering

- **Problem**: false triggering
  - Packets intended for one client may trigger other clients
  - Waste of energy

- **Solution**: PHY-layer addressing
  - Use sequence separation as node address

Node 0:

Node 1:

Sequence separation of 1
Addressing overhead

- **Problem:**
  - Preamble length ∞ number of addresses

- **Solution:** *minimum-cost address sharing*
  
  - Allow multiple nodes to be assigned the same address
  
  - Address allocated according to channel usage:
    
    * Clients with heavy channel usage *share* address *less* with others

  - Formulated as an integer program and solved via approximation
Switching overhead

- Delay caused by clock-rate switching

![Diagram showing packet transmission during clock-rate switching]

- **Problem**: how to prevent/minimize outage?

- **Solution**: Opportunistic Downclocking (ODoc)
  - Downclock the radio if there is *unlikely* to be any packet arrival within the switching time
  - How do we know this?

Switching period (9.5~151 µs)
Opportunistic Downclocking (ODoc)

- **Separate** deterministic packet arrivals $\implies$ no downclocking,
e.g., RTS $\rightarrow$ CTS $\rightarrow$ DATA $\rightarrow$ ACK

- **Predict** outage caused by non-deterministic packet arrivals
  - History-based prediction
    
    * History=1: outage occurs
    * History=0: otherwise
    * Next=1 if history contains 1 $\implies$ no downclocking
Integrating E-MiLi with sleep scheduling protocols

- State machine

- Add a new state downclocked IL (dIL)
- TX/RX $\leftrightarrow$ Sleep are managed by sleep scheduling
- SRID manages carrier sensing and packet detection
- ODoc determines whether and when to transit to IL or dIL
Deployment of E-MiLi

- **Coexistence with legacy WiFi**
  - E-MiLi nodes can detect legacy nodes via energy detection (or by directly detecting the legacy preamble)
  - Legacy nodes can detect E-MiLi nodes via energy detection (or by detecting a broadcast preamble)

- **Virtual carrier sensing**
  - Not supported in initial E-MiLi, but can be enabled by using `pkt duration` in an additional field preceding data pkt.
  - Only useful for preventing hidden terminal problem, but hidden terminal is not common in practice => turned off by default

Deployment of E-MiLi (cont’d)

- **Modifications** needed to enable E-MiLi
  - **AP:**
    * Firmware: prepending M-preamble
    * Driver: physical address allocation algorithm
  - **Mobile clients:**
    * Firmware: SRID algorithm
    * Driver: receiving allocated address

- **Commercialization:** licensed to world #1 smartphone manufacturer
Evaluation

- Packet detection
  - Software radio (PHY) based experiments

- Energy consumption
  - Packet traces from real-world WiFi networks

- Simulation for a wide range traffic patterns
  - Using ns-2
Packet-detection performance

- Single link
  - Use USRP nodes and vary SNR and clock-rates
- Multiple clients and 1 AP
  - Lab/office environment
  - All except D are stationary

- Detection performance
Energy savings

- Trace-based simulation for networks of 30+ and 7 users
  - Based on WiFi power profile (Max downclocking factor 4)

\[ 40\% \text{ of energy savings!} \]
Simulation of synthetic traffic

- Implementation in ns-2
  - MAC layer: ODoc
  - Switching delay: $151 \mu s$ (the worst case)
  - SNR: 8dB (pessimistic)
- Performance of a 5-minute Web browsing session:
  >40% energy savings with negligible delay overhead
Performance when downloading a 20MB file using FTP:
>40% energy savings with 5% throughput loss
Effects of E-MiLi overhead

- **M-preamble** and **switching delay** costs extra channel time
- Overhead/data **ratio** increases as data rate increases
- This effect is negligible for: PSM+E-MiLi; devices with short switching delay
2\textsuperscript{nd} Solution: Gap Sense

- Principles of Gap Sense (GSense):
  - Prepend a \textit{modulation-agnostic} preamble to each packet
  - Preamble comprises \textit{multiple energy pulses}
  - Gap duration between pulses is used to convey coordination information between heterogeneous devices
Preamble design [1/3]

- Energy pulses, each containing a sequence of digital samples
  - Should be sufficiently long to be detectable by heterogeneous rx’s
    * Depends on bandwidth heterogeneity
    * If preamble transmitter’s BW is $D$ times the receiver’s BW, then pulse length must be $\geq D$ digital samples
  - But should not be too long to prevent the receiver’s confusion of it as a regular pkt => should be shorter than minimum of regular pkt durations in coexisting networks
    * Should be shorter than minimum of header durations in coexisting networks
      * E.g., 160us for bluetooth, 100us for Zigbee, 8us for WiFi
Preamble design [2/3]

- Gap length between energy pulses

  - Should not exceed the carrier sensing duration of ambient devices, e.g., 128us for ZigBee, 28-50us for WiFi

  - Otherwise, those devices may start transmission
Random or deterministic sequence?

- Traditionally, communication sequences are *random*
  - A *scrambler* is used to randomize bit sequences
  - Avoiding long sequences of 0’s or 1’s, to facilitate error control code

- GSense’s preamble sequence
  - Objective: *maximize* energy pulses’ RSS at receiver, i.e., detection prob
    - => No need for error control
  - *Deterministic sequence* results in higher RSS than random sequence
    since in *random sequence*, adjacent samples weaken each other
Embedding coordination information into preamble

- **Gap duration** as coordination information
  - # of 0 samples within a gap
  - Translated directly into a decimal number
  - Decimal number is then mapped to **coordination information**
    - Mapping should be *customized* by protocols that use GSense
  - Multiple energy pulses can be concatenated, each gap representing **one field** of info
  - Only suitable for *low-rate* control & coordination since use of gap to convey info is less efficient than modulation schemes.
Preamble detection

- Heterogeneous networks should agree on common parameters offline
  - e.g., length of energy pulses, bandwidth heterogeneity ($D$), map
    between gap length and coordination information

- Detection steps: for each incoming digital sample,
  - Compute energy level for each sample => smooth over recent samples
    => collect sufficiently high SNR samples to decide on an energy pulse
    => determine gap length

![Graph showing Preamble detection process](image)
Accuracy of preamble detection

- Experimental setup
  - Implementation on GNURadio/USRP2
  - Benchmark: E-MiLi, self-correlation-based preamble detection

* Error rate is close to 0 in a common (above 8dB) SNR range
Effect of sequence design

- Random vs. deterministic sequence for different values of $D$

* At low SNR, deterministic sequence can reduce mis-detection prob by multiple folds
Accuracy of detecting gap information

- Error prob while varying $D$

* At low SNR, larger $D \Rightarrow$ higher error prob
* Error prob close to 0 when SNR is above 8 dB
Evaluation of GSense

- Experimental setup
  * Simulation with real WiFi packet traces
  * Benchmark: E-MiLi, self-correlation-based packet detection algorithm

* GSense’s energy savings is comparable to E-MiLi: 44% for more than 90% clients
Effects of data-rate

- Experimental setup
  - Packet-level simulation
  - Benchmark: E-MiLi

* GSense’s energy savings (23.1 -- 44%) and throughput loss are comparable to E-MiLi, under various data-rates
Additional advantages over E-MiLi

- **Lower-complexity**: energy detection vs. self-correlation-based detection

- **Better compatibility** with legacy hardware since energy sensing is a common built-in function in wireless devices

- **Easier realization of virtual carrier sensing**
  
  * Hidden terminal can sense CTS by sensing its gap duration
  * E-MiLi needs to prepend a randomized preamble => modifications to legacy hardware/firmware
GSense Applications

- GSense as a **general coordination** mechanism
  - Identify problems in heterogeneous networks
  - Identify necessary coordination information
  - Define **mapping** between gap length and coordination information

- Applications
  - Reducing collision btw ZigBee and WiFi
  - Reducing collision btw WiFi networks of heterogeneous bandwidths
  - Coordinating tx and rx with different clock-rates to save energy
Current Status of GSense

• US patent-pending
• Under discussion for potential licensing
• google “Gap Sense” to find press coverage
Conclusion

- Idle Listening (IL) dominates WiFi energy consumption

- E-MiLi reduces IL power by adaptive clock-rate
  - Separate packet detection from packet reception
  - SRID: detects packets at low clock-rate
  - ODoc: integrates E-MiLi with MAC-layer sleep scheduler

- GSense: a general, lightweight coordination mechanism
  - A simple coordination mechanism for heterogeneous networks
  - Using energy pulses & gaps to convey information
  - High detection accuracy in common SNR ranges
Thank you!