Surviving Wi-Fi Interference in Low Power ZigBee Networks

Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu,

Andreas Terzis, SenSys 2010

Simon Gerber
simugerber@student.ethz.ch

Mentor: Philipp Sommer

23.03.2011

Motivation

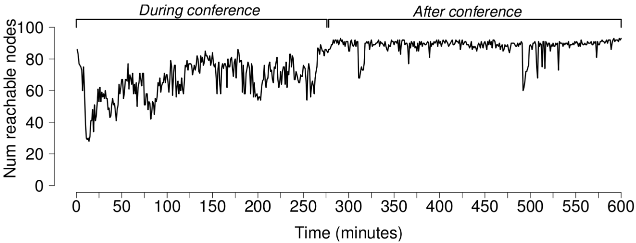
- More and more wireless technologies deployed
- Many in the 2.4GHz ISM band
- Cross Technology Interference is becoming a problem

Cross Technology Interference

- Interference from other wireless technologies considered the same as random background noise in most MAC protocols
- Especially a problem for 802.15.4 (ZigBee) networks in the presence of WiFi

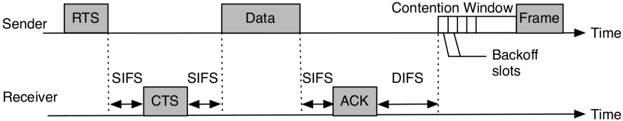
A representative experiment

- 90 sensor nodes in a 13,000 m^2 ($\approx$ 1.8 football fields) lecture hall using four 15.4 channels
- Co-located WiFi network which uses all channels across the entire space
- During Microsoft PDC conference more than 2500 people connected to WiFi network



WiFi (IEEE 802.11{b,g})

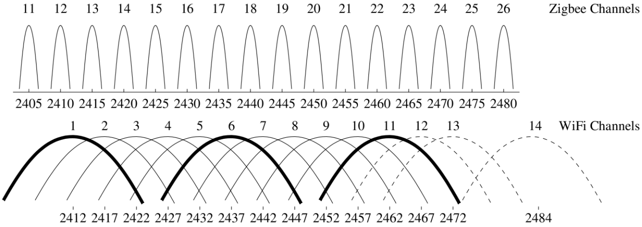
- 802.11 specifies CSMA/CA with ACKs for channel access
- Optionally also RTS/CTS packets
- SIFS and DIFS intervals
- Main difference between 802.11b and 802.11g: timing of SIFS/DIFS/slot length
- Transmission power in the order of 100 mW
- Packet length 194 µs – 542 µs (for 802.11g)



ZigBee (IEEE 802.15.4)

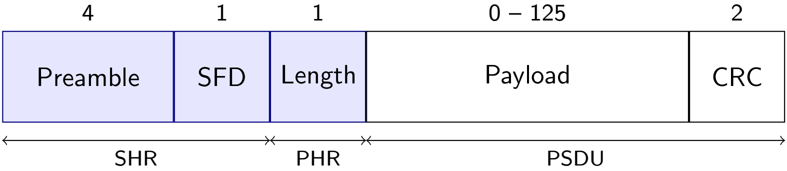
- IEEE 802.15.4 defines a PHY layer for low-rate wireless networks in the 2.4 GHz ISM band
- 16 channels within band, each 2MHz wide with 3MHz inter-channel gap-bands
- Outgoing bytes are divided into 4 bit symbols
- Each symbol is mapped to one of 16 pseudo-random, 32 chip sequences
- Radio uses O-QPSK encoding and transmits at 2 MChips/s (250 kbps)
- Transmission power usually 1 mW
- Packet length 352 μs – 4256 μs

ZigBee and WiFi channels



- Most WiFi networks use channels 1, 6, or 11

ZigBee packet format



4	1	1	0 – 125	2
Preamble	SFD	Length	Payload	CRC
SHR		PHR	PSDU	

- 5 byte synchronisation header (SHR)
- 4 byte preamble, all bytes set to `0x00`
- 1 byte start of frame delimiter set to `0x7A`
- 1 byte PHY header (PHR)
- 1 byte length field containing number of bytes in the packet including 2 byte CRC

Measuring ZigBee performance

Ko, Gao, and Terzis: Empirical Study of a Medical Sensor Application in an Urban Emergency Department, *BodyNets 2009*

- Empirical Results in a hospital setting
- End-to-end packet throughput of a 15.4 network overlapping a 802.11 network decreases by a factor of three

Hauer, Handziski and Wolisz: Experimental Study of the Impact of WLAN Interference on IEEE 802.15.4 Body Area Networks, *EWSN 2009*

- Positions of bit errors in 15.4 packets are temporally correlated with WiFi traffic

Improving ZigBee performance

Musaloiu-E and Terzis: Minimising the Effect of WiFi Interference in 802.15.4 Wireless Sensor Networks, *International Journal of Sensor Networks, 3(1):43–54, 2007*

- Distributed Channel Selection Mechanism which detects WiFi interference

Srinivasan, Kazandjieva, Agarwal, and Levis: The β-Factor: Measuring Wireless Link Burstiness, *SenSys 2008*

- Off-line strategy to quantify the level of link burstiness due to interference
- Estimate expected duration of interference and defer packet transmissions

Improving WiFi performance

Han et. al: Maranello: Practical Partial Packet Recovery for 802.11, *NSDI 2010*

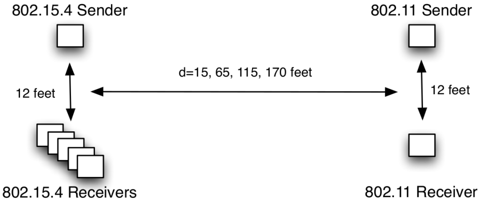
- Applying CRC on blocks of the WiFi payload

Jamieson and Balakrishnan: PPR: Partial Packet Recovery for Wireless Networks, *SIGCOMM 2007*

- Replicate Packet header at the end of the WiFi packet
- Does not work on existing hardware

Experiment Setup

- Basement (very low outside interference)
- WiFi: one laptop and one access point
- ZigBee: one sender, five receivers
- Experiment run for d = 15/65/115/170 feet
- Each time with b and g WiFi
- WiFi sender generates a stream of 1500 byte TCP packets
- ZigBee sender sends one packet w/ 128 bytes payload every 75 ms

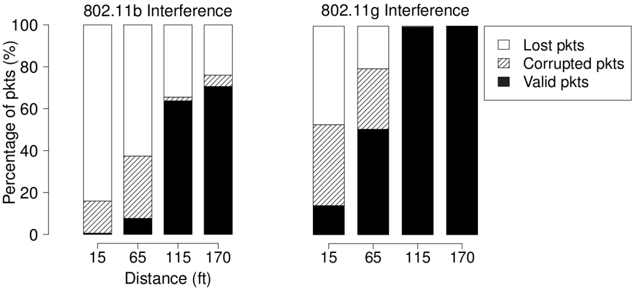


Methodology

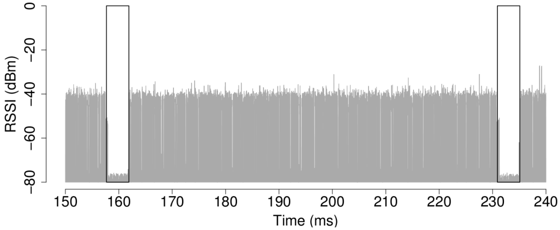
- Previous work mostly focused on high level interactions: e.g. packet reception ratio
- Interaction between WiFi and ZigBee examined by accurately measuring packet transmission events
- The radio used for measuring generates an analog voltage on its `RSSI_OUT` pin corresponding to the signal energy received in a 2MHz frequency band centered on the tuned frequency

Reception Ratio

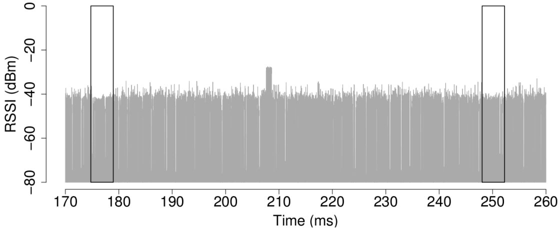
- 802.11b traffic has larger impact
- Front part of 15.4 packet more vulnerable
- Transmission latency increased
- Also TCP throughput on the WiFi network drops by 4% at $d = 15$ feet



Packet Transmission Timeline

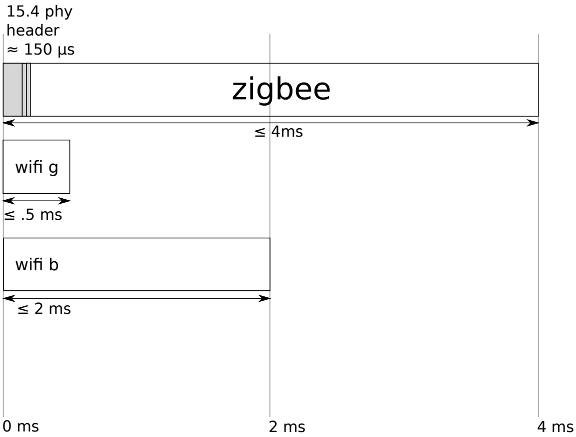


(a) $d = 15$ feet.



(b) $d = 115$ feet.

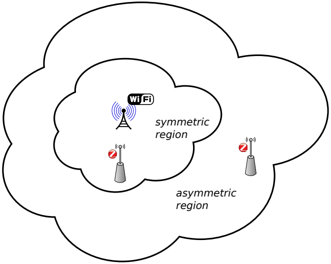
Packet Length Comparison



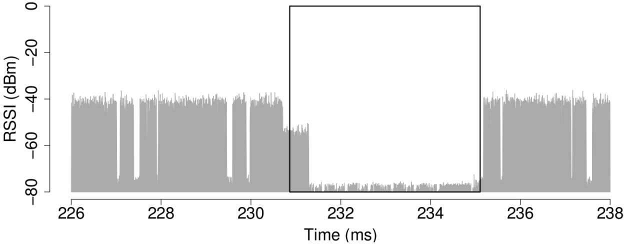
Interaction Dynamics

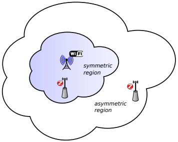
- 802.11 backs-off during 802.15.4 transmissions when the distance between 802.11 and 15.4 nodes is small
- Cause: CCA mandated by the 802.11 specification
- Not all 802.11 radios will back-off: those that do packet detection will declare the channel as clear
- This defines two interference regions:

  symmetric and **asymmetric**

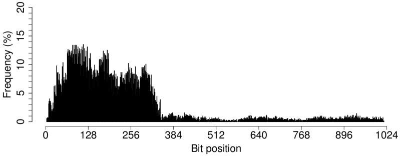


Packet Transmission Timeline: Detail

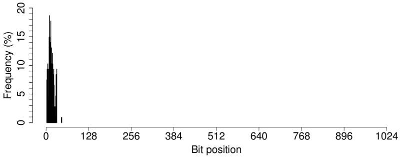




Bit Error Distribution: Symmetric Region

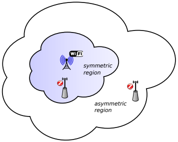


(a) 802.11b interfering source.

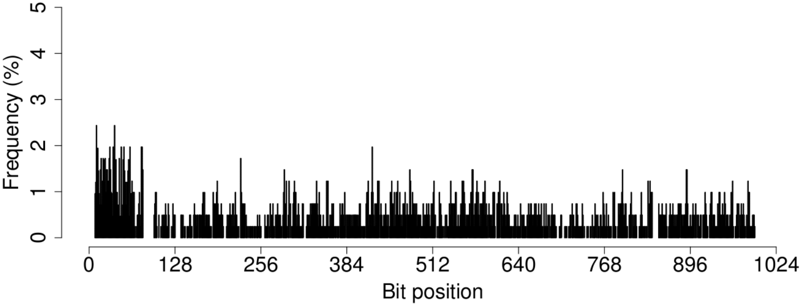


(b) 802.11g interfering source.

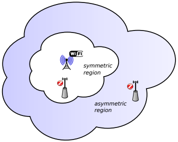
- Most bit errors in the front section



Bit Error Distribution: Asymmetric Region



- Errors distributed uniformly across the whole packet



Two Problems, One Solution

P1 In the symmetric region packets are not received due to corrupted headers

P2 In the asymmetric region received packets often have corrupted payloads

S1 Multi-Headers

S2 Forward Error Correction

- Combine S1 and S2 in a MAC-layer solution

BuzzBuzz

BuzzBuzz: Mode of Operation

- BuzzBuzz infers channel quality by observing packet losses or the lack of acknowledgements
- The sender first tries to deliver packets using ARQ
- After three unsuccessful attempts delivering the packet, the FEC information is added and one MH header is inserted
- After another three unsuccessful attempts the sender gives up on that packet

Why BuzzBuzz belongs in the MAC-layer

- MAC typically maintains neighborhood and link quality information
- In the MAC layer the underlying radio header format is known
- Running FEC for every hop eliminates accumulations of bit errors

Multi-Headers (MH): Design

4	1	1	10	4	1	1	10	0 – 115-16N	2
0x00	0x7A	Len	MAC	0x00	0x7A	Len	MAC	Payload	CRC
Original Header				MH Header					

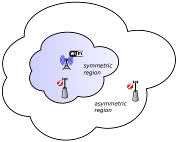
- Light-weight, sender-initiated
- Similar to having longer preambles
- Add multiple headers to a packet
- Adjust length field according to the number of headers after the current one
- Need to disable hardware CRC

Multi-Headers: Effectiveness

- Five 802.11g clients connected to AP
- 15.4 network in same office as 802.11 clients and AP
- 15.4 sender 15 feet away from four 15.4 receivers

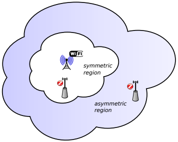
WiFi traffic	**15.4 Header**	**Additional Headers**		
type		1st	2nd	3rd
TCP	30.5%	49.5%	10.0%	1.9%
UDP	28.2%	53.9%	12.9%	1.8%

Percentage of packets successfully received using the original or one of the additional headers



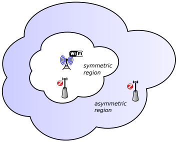
Possible Methods in the Asymmetric Region

- Packet Retransmission
- Forward Error Correction



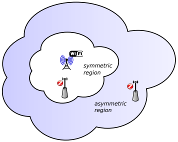
Error-Correction Codes

- Transform message to larger encoded message
- Redundant information in encoded message allows receiver to recover a limited amount of bit errors



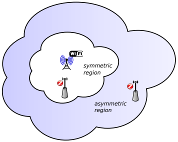
Hamming Code

- Technique: Add extra parity bits to the message
- Each parity bit enables detection of up to two bit errors
- Each parity bit enables correction of one bit error



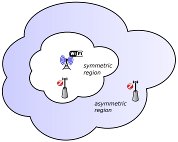
Hamming Code

- Effectiveness tested with Hamming(12, 8) code
- Adds 4 parity bits to 8 data bits
- Can detect and correct one bit error in the 12 bit code word
- To verify correctness of a decoded message with a unknown number of bit errors other techniques such as a CRC code need to be used



Hamming Code: Implementation

- 72-byte messages which are encoded using Hamming(12,8) to 108-byte encoded messages
- Two 12-bit code words are packed into three bytes
- The 72-byte message contains a 2-byte CRC to verify correctness
- Takes 1.4 ms and 1.8ms respectively to encode and decode a 108-byte message on a TelosB mote (4 MHz)

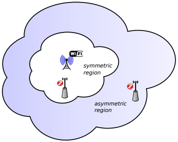


Hamming Code: Evaluation

Percentage of corrupted payloads that can be recovered

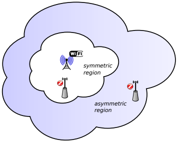
	Hamming(12, 8)		*Hamming(12, 8) w/ Bit Interleaving*	
	11b	11g	11b	11g
15 ft	0.6%	11.7%	12.4%	57.6%
65 ft	4.7%	19.1%	55.6%	70.4%

- Applying bit interleaving gives much better recovery rates
- Bit interleaving is done in such a way that two consecutive bits in a 12-bit code word are separated by 72 bits



Reed-Solomon Code

- Block based
- Can recover from data corruptions and erasures
- Divides message into x blocks of user-defined size
- Computes a parity of y blocks



Reed-Solomon Code: Recovery

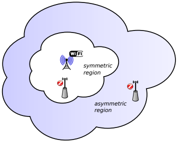
- Encoded message consists of original message and computed parity
- For y blocks of parity RS can recover from:

$$2 \times (num\_corrupted\_blocks) + 1 \times (num\_erasure\_blocks) < y$$

Reed-Solomon Code: TinyRS

- Full-featured TinyOS compatible RS library
- 8-bit block size and 30-byte parity
- Micro-benchmark with message payload of 65 bytes

Encoding	**Decoding**		
	15-byte error	30-byte erasure	no errors
36.156 ms	181.892 ms	207.824 ms	104.296 ms

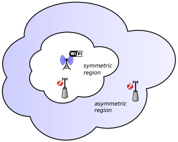


Reed-Solomon Code: TinyRS Evaluation

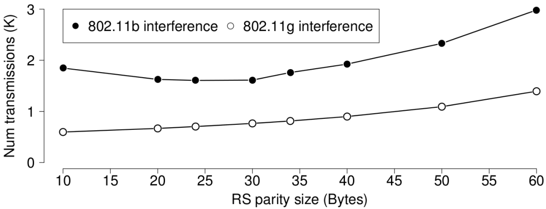
Percentage of corrupted packet payloads that can be recovered

	Hamming(12, 8)		*Hamming(12, 8)*		*RS*	
			w/ Bit Interleaving		*w/ 30-byte parity*	
	11b	11g	11b	11g	11b	11g
15 ft	0.6%	11.7%	12.4%	57.6%	52.0%	65.2%
65 ft	4.7%	19.1%	55.6%	70.4%	85.3%	85.9%

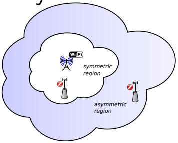
RS can successfully recover four times more packets than Hamming(12,8) w/ bit interleaving when packets are corrupted by a 802.11b transmitter in the symmetric region



Reed-Solomon Code: Parity Size

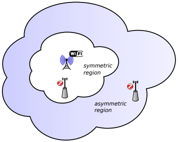


- Simulation to determine expected number of transmissions necessary to deliver a 38 KB object
- The result of this simulation suggests that 30-byte parity requires the smallest number of transmissions



Reed-Solomon Code: TinyRS Effectiveness and Power Consumption

- Comparing RS FEC with other packet recovery strategies
- Packet-level and block-level ARQ
- Both rely on acknowledgements to decide whether to retransmit
- To simplify the comparison, assume all acknowledgements are delivered

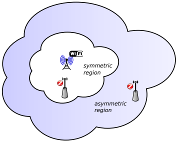


Reed-Solomon Code: TinyRS Effectiveness and Power Consumption

Delivering a 38 KB object

Method	number of packets	energy mA s
Packet-level ARQ	4,409	1,290
Block-level ARQ (30-byte blocks)	2,313	284
TinyRS (30-byte parity)	1,720	748

Considering lost acknowledgements, ARQ would use even more energy



BuzzBuzz: Evaluation Setup

- 57-node TelosB testbed deployed in an office building
- Benchmark Data Collection in WSN using the Collection Tree Protocol (CTP) to deliver 65-byte application data from each node at a rate of one packet per minute
- 802.11 interference generated by OpenMesh ad-hoc mesh backbone and three N800 internet tablets generating traffic
- 802.11 traffic started 20 minutes after starting 15.4 nodes to ensure CTP had sufficient time to build its routing tree

BuzzBuzz: Evaluation Results

	CTP	CTP w/ BuzzBuzz
Packet Delivery Rate	43.05%	73.90%
Avg. Number pkts/s in the network	38	11
% pkts not ACKed	66%	35%
% pkts received due to MH hdr	N/A	10.58%
% corrupted pkts recovered with RS	N/A	42.69%
% decrease in 802.11g throughput	14.51%	3.35%

Future Work: Network-wide blocker

- BuzzBuzz is a reactive approach
- Each node operates independently to mitigate WiFi interference
- Possible to design proactive solutions for dense ZigBee networks
- Use a collection of dedicated 802.11 blockers placed close to each 802.11 node
- Simple experiment with one blocker next to the 802.11 AP shows an increase of 26% in 15.4 throughput

Conclusions

+ 70% increase in PRR

+ Number of 15.4 transmission reduced by a factor of 3

+ BuzzBuzz adapts to the amount of channel noise

– RS overhead might be prohibitive in terms of power consumption (3 times the energy of block-level ARQ)

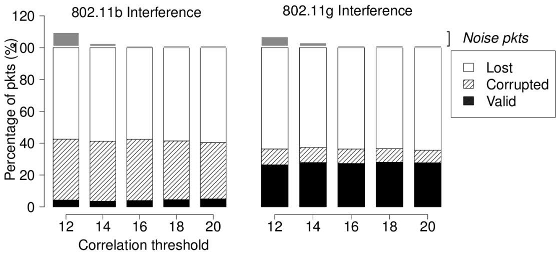
– Much focus on 802.11b

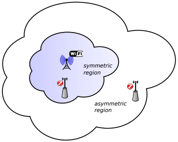
Any questions

?

Correlation Threshold

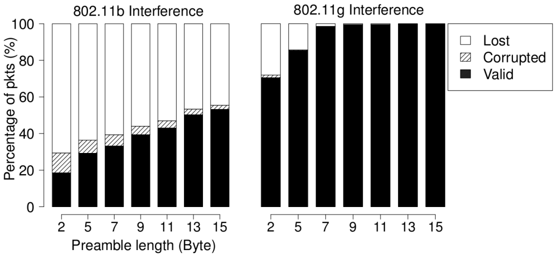
- Some 15.4 radios provide a configurable correlation threshold
- The threshold determines the amount of noise that is tolerated decoding chip sequences when searching for the SHR

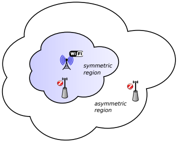




Preamble Length

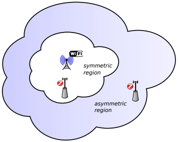
- The standard specifies a 4 byte preamble
- Some radios allow the user to set the length of the preamble
- Upper limit of preamble length mandated by hardware





ECC example

- Linear Code in the 15.4 PHY layer
- Map 4-bit symbols onto 32-bit chip sequences
- Minimum Hamming distance between any two of the 16 predefined chip sequences is 12
- Chip sequences containing no more than 6 bit errors can be mapped to the correct 4-bit symbol



Performance under 802.11n interference

- The 802.11n standard introduces several new features
- These features do not completely mitigate the CTI problem between 15.4 and 802.11 networks
- The bit-rate increase from 802.11g to 802.11n does not increase 15.4 PRR by much (3%) compared to the increase from 802.11b to 802.11g (up to 700%)
- Channel bonding is a mechanism introduced by 802.11n which makes it even more difficult to find an interference-free 15.4 channel