

DOT3 Radio Stack

Jaein Jeong, Sukun Kim

Nest Retreat

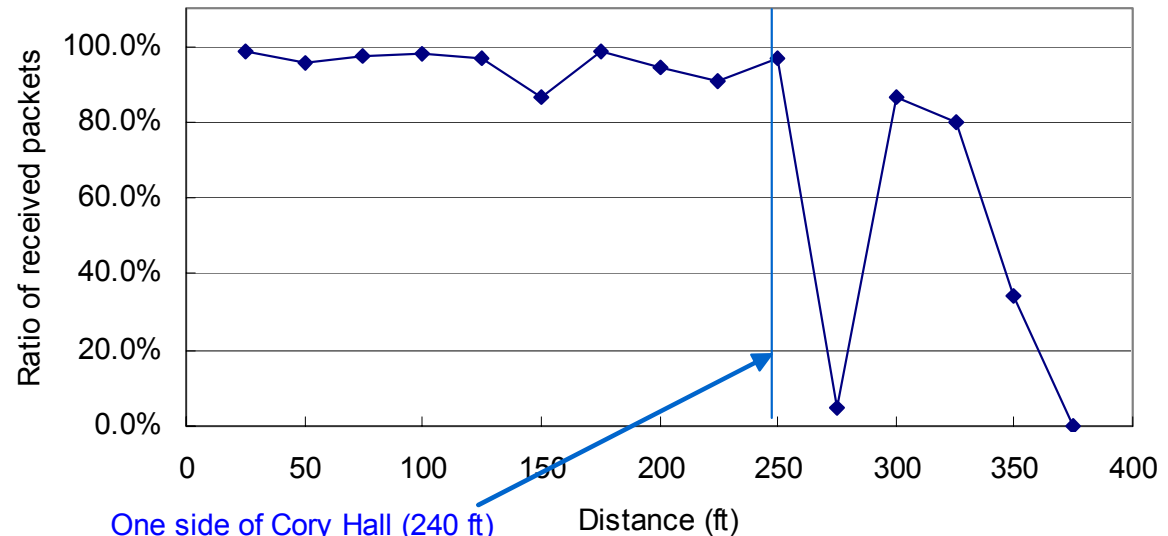
January 16, 2003

Introduction

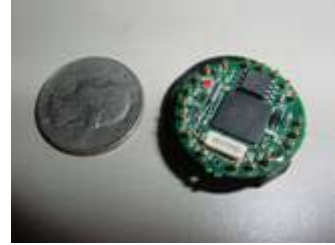


- **A wireless sensor**
 - sample analog signals
 - communicate with other nodes in wireless.
- **MICA is the current platform in Berkeley.**
- **MICA has been useful, but not enough for large scale app due to short range**

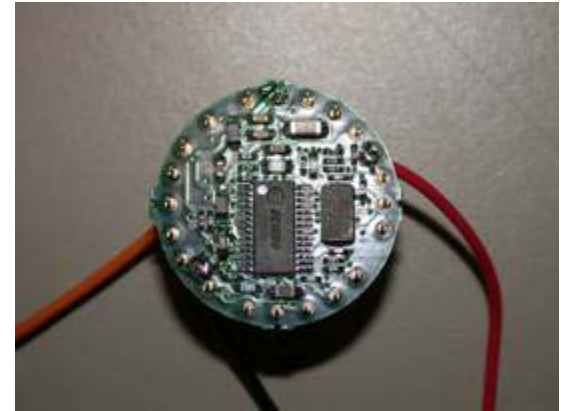
Outdoor range of MICA (256 packets)



Mote with CC1000 Radio



- DOT3 is a new platform with ChipCon CC1000 radio chip.
- MICA2 is a variation of DOT3 that has full features of MICA.
- We aim to have a working network stack for motes with ChipCon radio in nesC.



A DOT3 with its radio chip in the middle

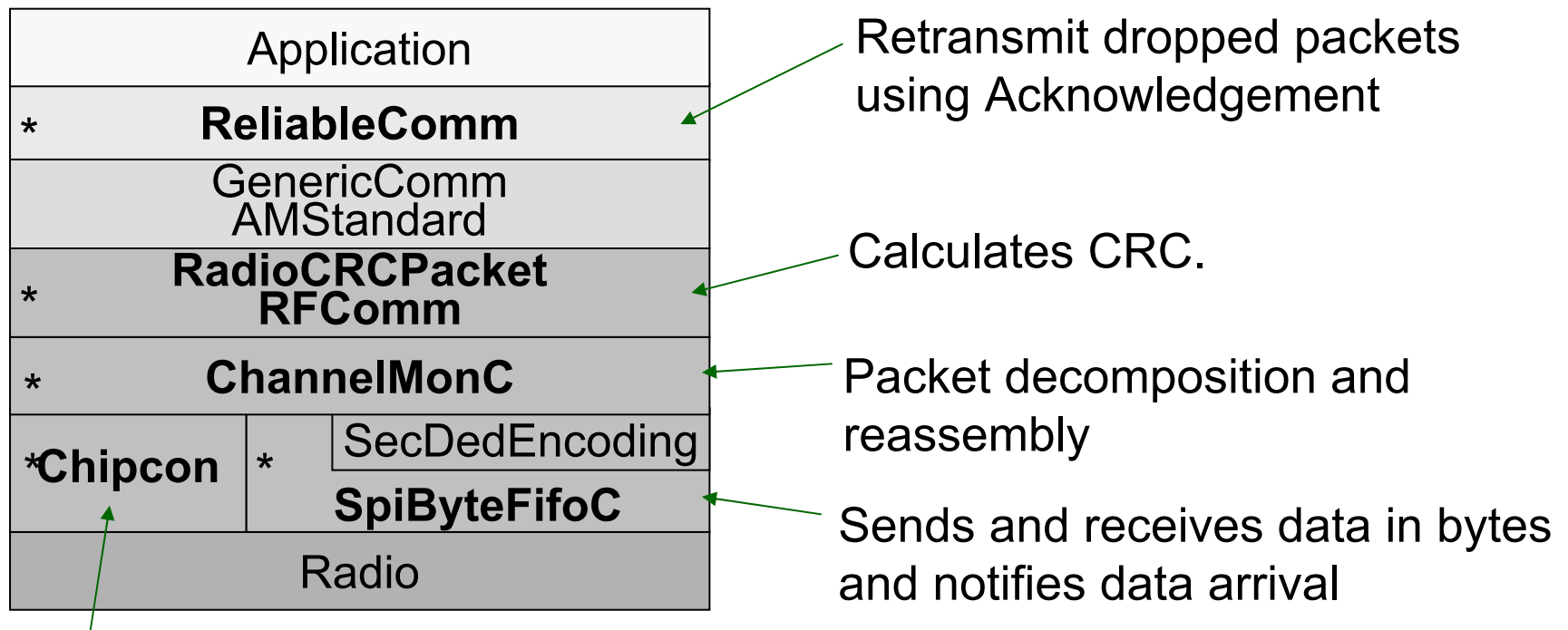


A MICA2 mote

Design of Chipcon Radio Stack



- Components accessing the radio were modified
- Components for reliable communication were added.

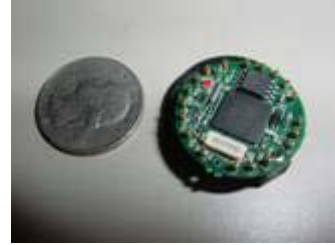


Packet decomposition and reassembly

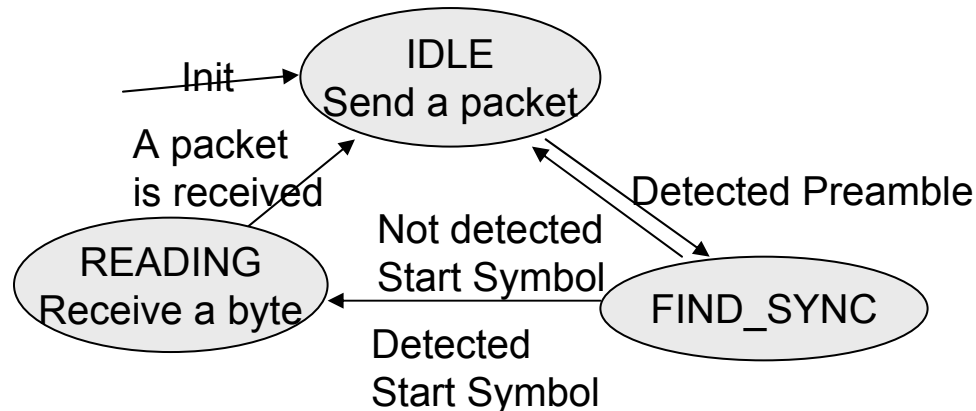


- The application level uses a packet whereas the underlying radio uses a byte as a data unit.
- Thus, a packet needs to be decomposed to bytes and reassembled from bytes.
- Since a packet is received as a sequence of bytes, we need a way to tell the beginning of the packet.
- The byte data can be transferred in half duplex mode.

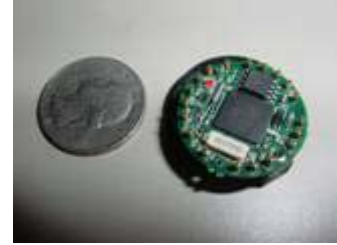
Packet decomposition and reassembly



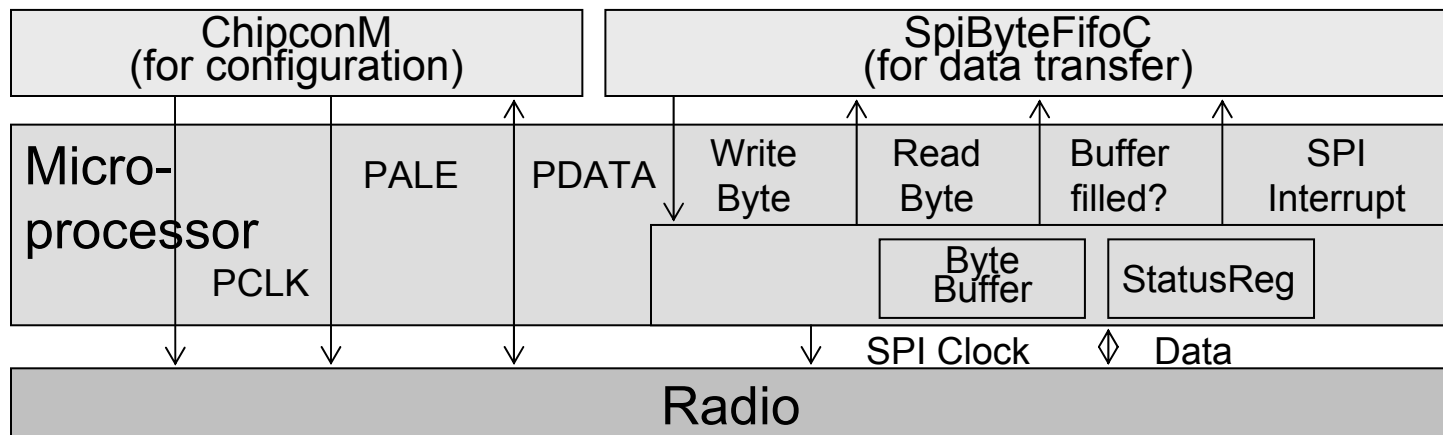
- Packet decomposition and reassembly can be implemented using a state machine in the below:
- Send mode consists of one state
 - IDLE state: sends a byte when the byte buffer is empty
- Receive mode consists of two states
 - FIND_SYNC state : detects the start of a packet using preamble and start symbol
 - READING state: reads the remaining bytes and triggers an event when all the bytes are read.



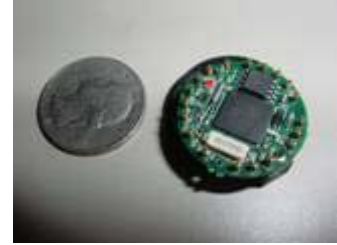
Interface to CC1000



- **Microprocessor transfers data to and from the radio using byte level interface called SPI.**
- **The microprocessor needs to communicate with CC1000 radio chip to configure or monitor the status of it.**
- **The properties like operating frequency and power consumption can be set up by changing the CC1000 status registers.**



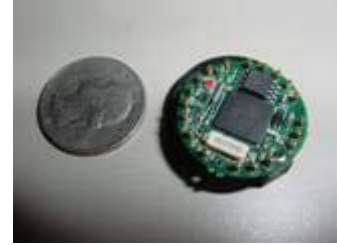
Using multiple channels



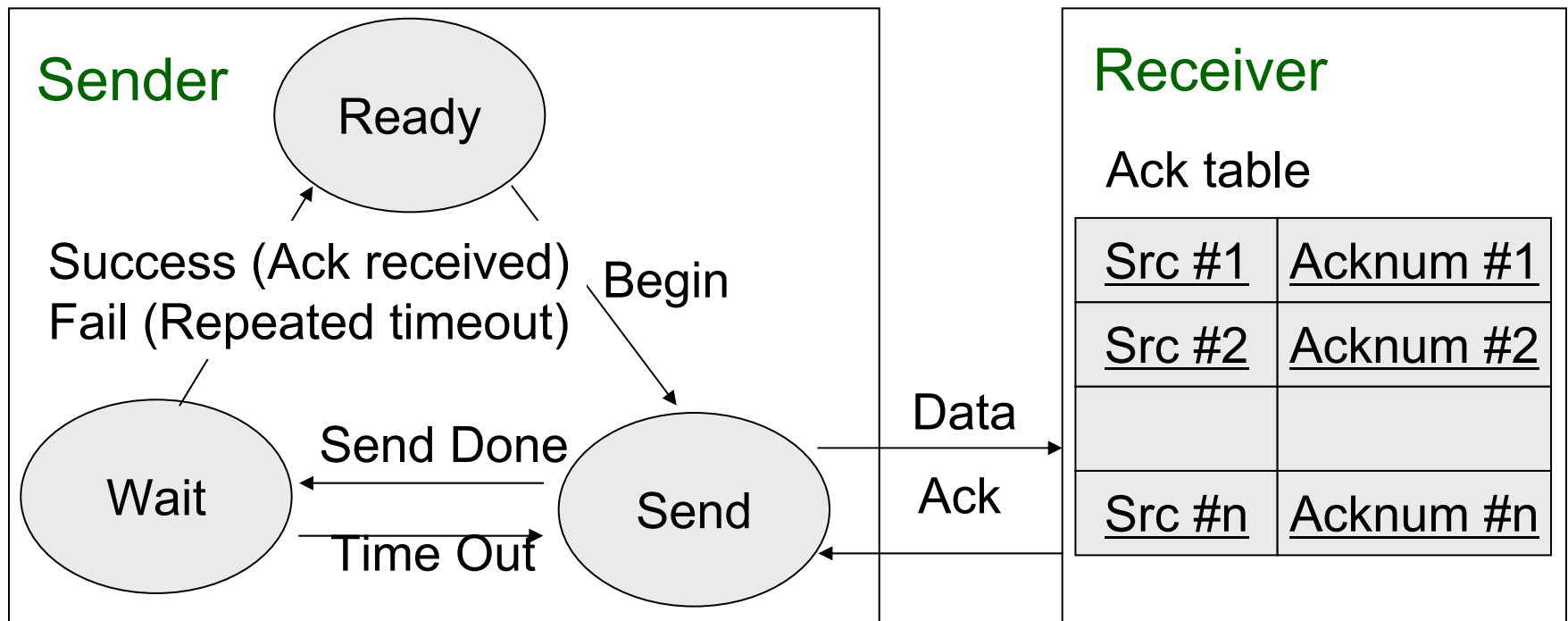
- **CC1000 can operate in several different bands: 433, 866 and 916 MHz using corresponding capacitors and inductors.**
- **Within each band, CC1000 can operate in different frequencies according to the status register values.**
- **Using multiple channels can help reducing the interference between nodes.**
- **We found working frequencies in 433 MHz band and here are the examples:**

		CH 1	CH 2	CH 3	CH 4	MICA
TX	Frequency(MHz)	433.02	433.64	434.20	434.71	916.50
RX	Frequency (MHz)	433.09	433.71	434.27	434.78	916.50

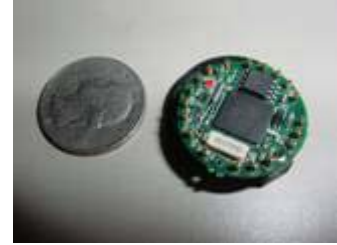
How to transmit messages reliably?



- Add source address and Ack number to packets.
- Receiver keeps track of senders to handle duplicate packets

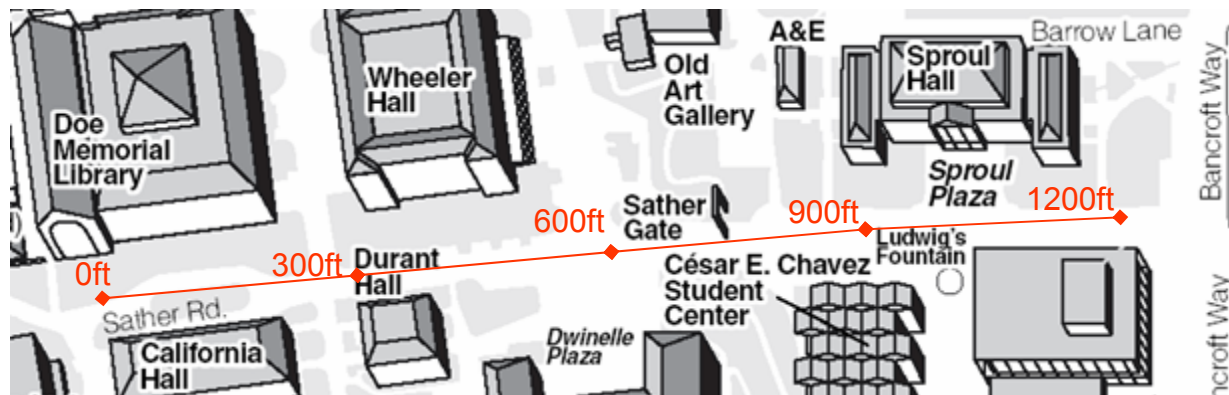


Evaluation

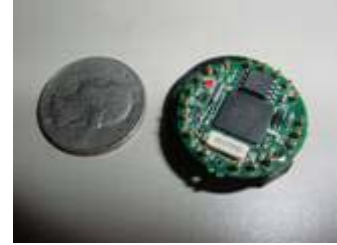


- **Evaluation Methods**

- Sends a number of packets and counts the packets received as we vary the environment.
- Ratio of received packets is our metric.
- In outdoor tests, we vary the distance.
- In indoor tests, we vary the number of nodes and number of channels used.

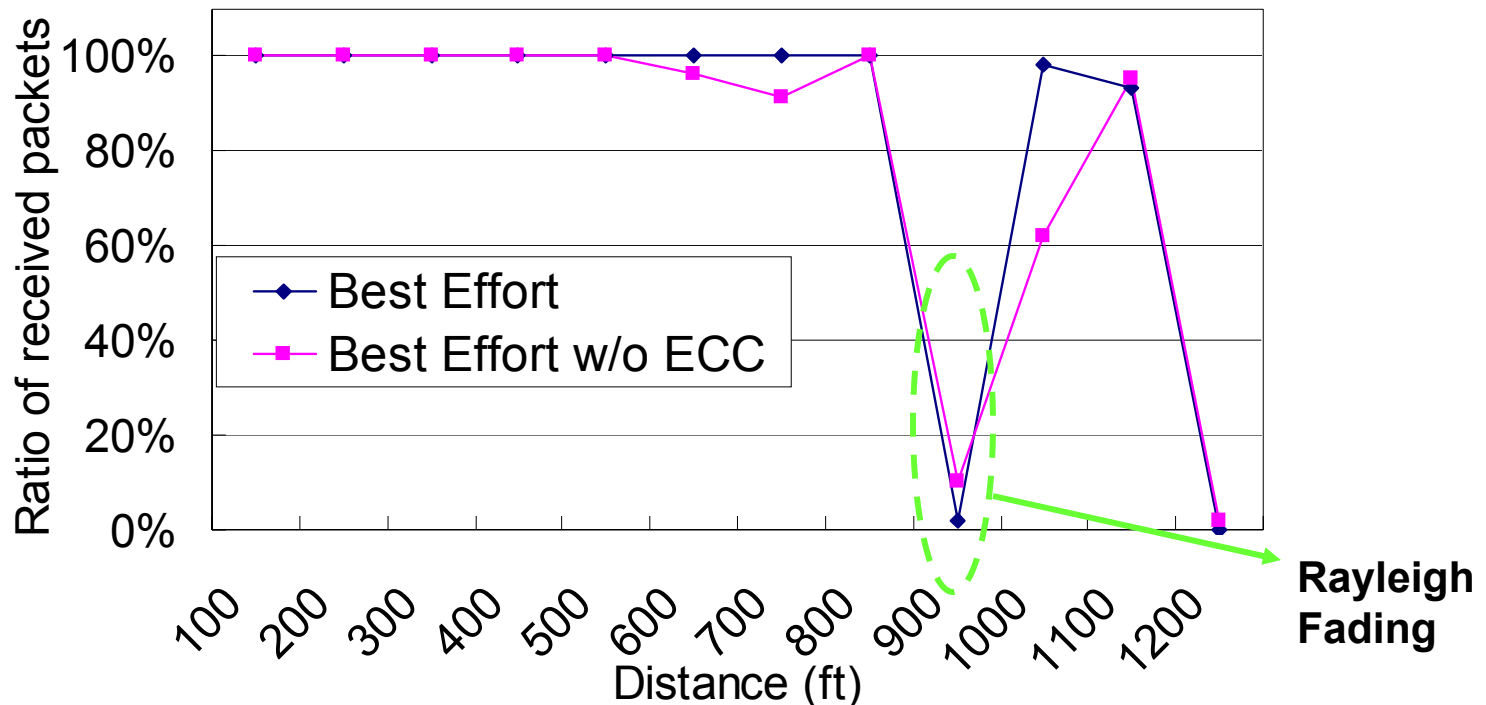


Effectiveness of ECC

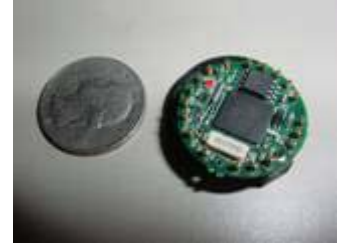


- **Transmission with error correction code, no packets were dropped within 800ft compared to 500ft for non-ECC version.**

The effectiveness of ECC (256 packets)

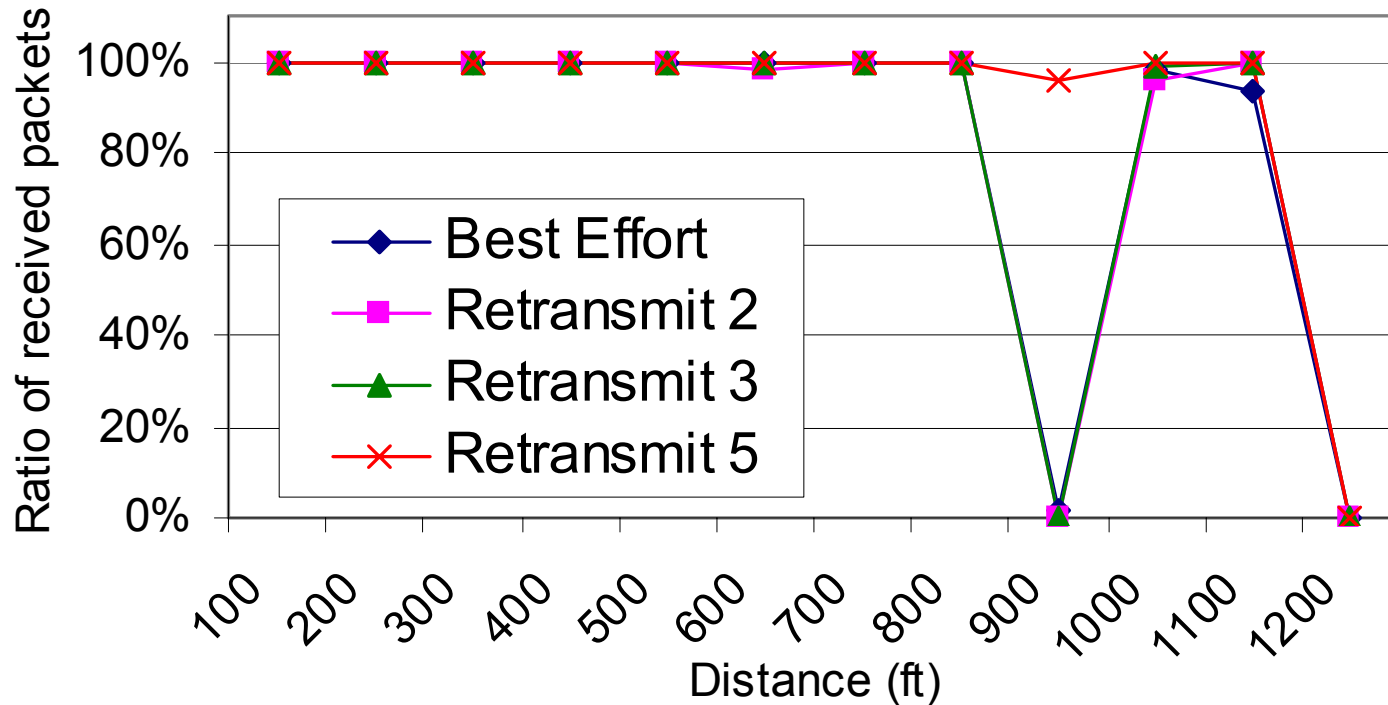


Effectiveness of retransmission

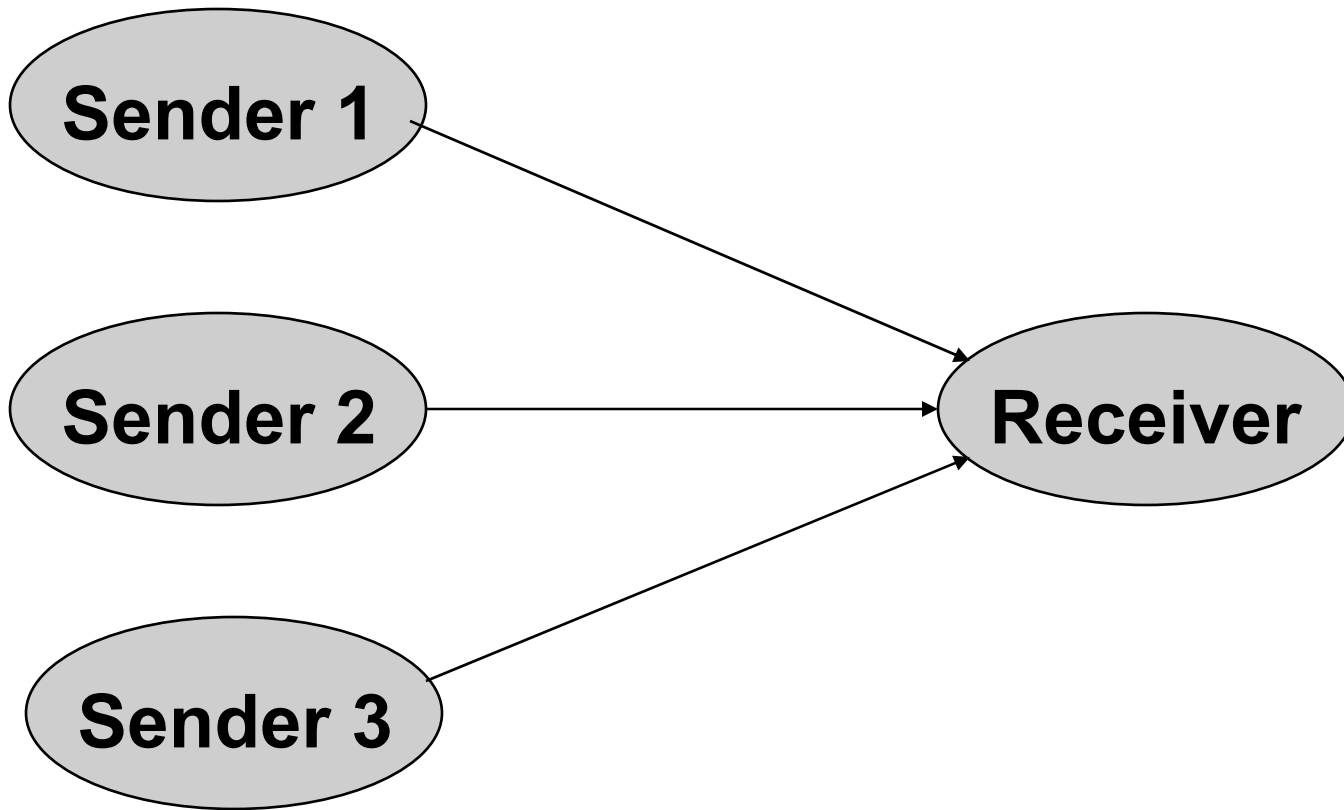


- **Retransmission reduced the packet losses with additional time costs.**

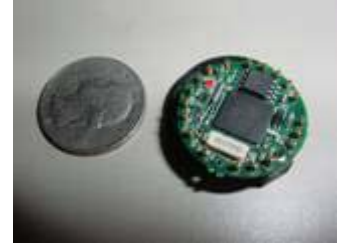
The effectiveness of retransmission (256 packets)



Multiple Senders

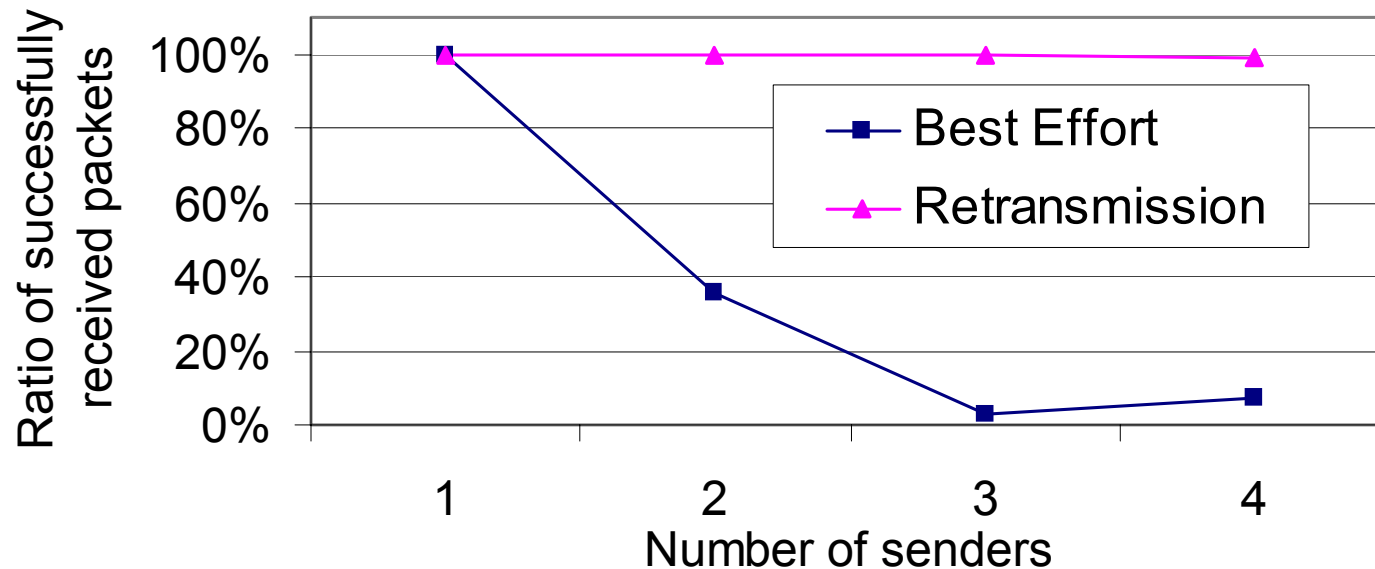


Cases with multiple senders

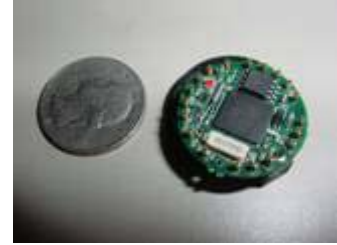


- **Retransmission reduced most of the packet losses due to collision.**

The effects of retransmission on collisions
(128 packets per node)

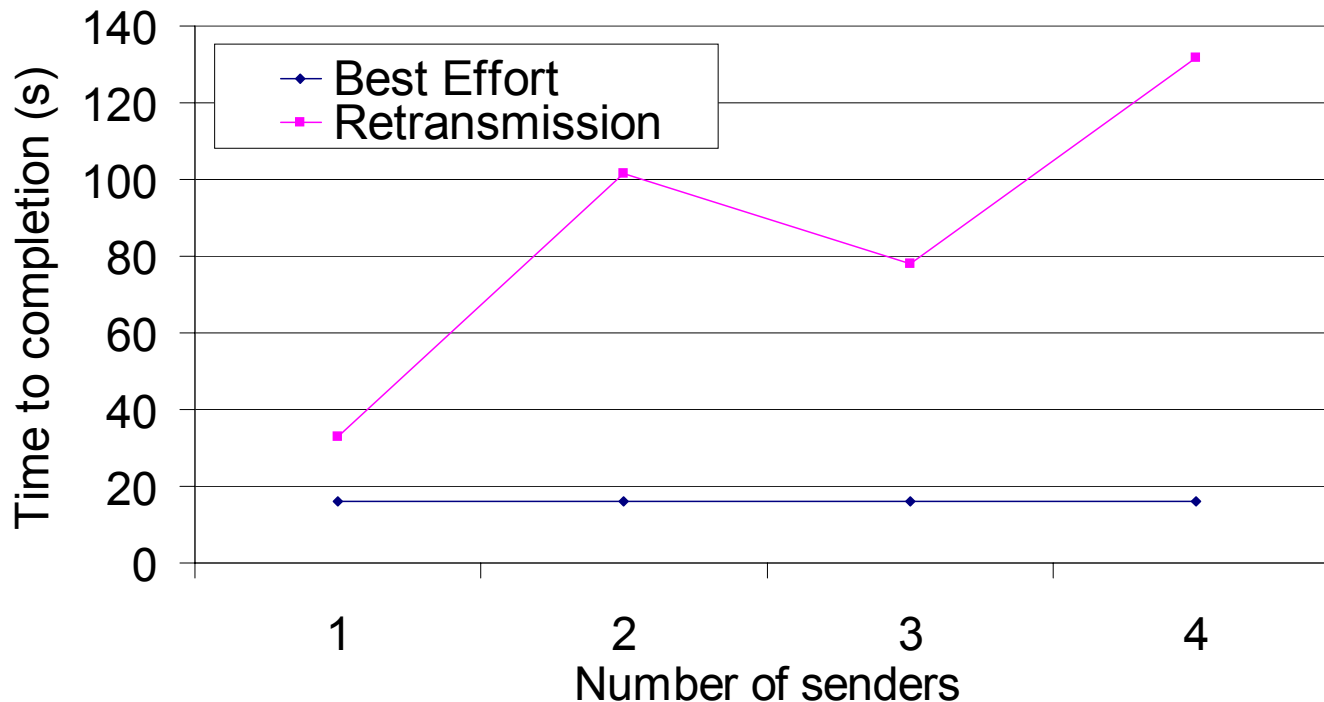


Cases with multiple senders

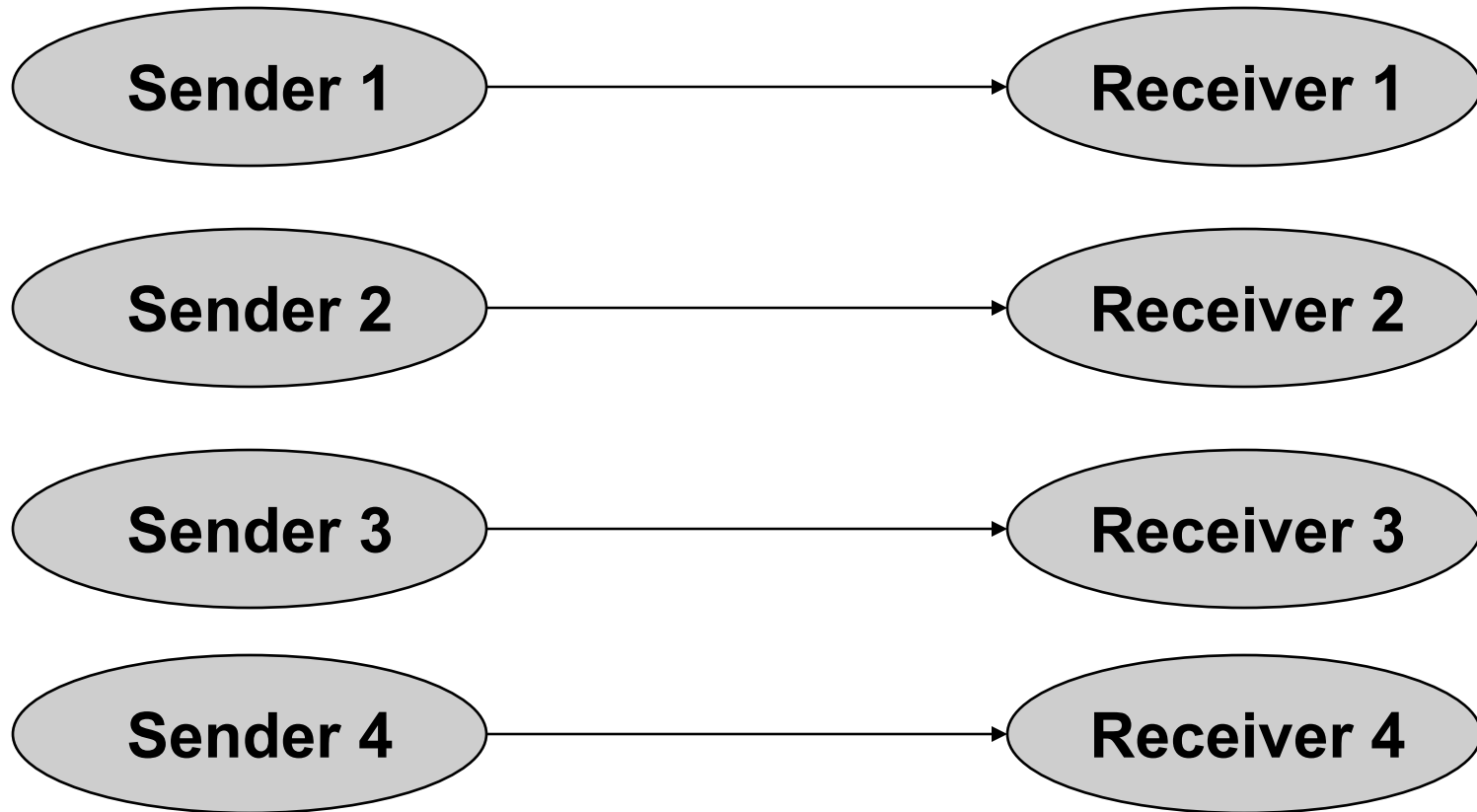
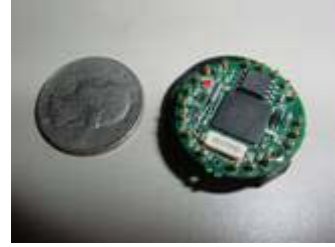


- **Retransmission paid a little high costs for increasing packet receiving rate (over 6 times in case of 4 senders).**

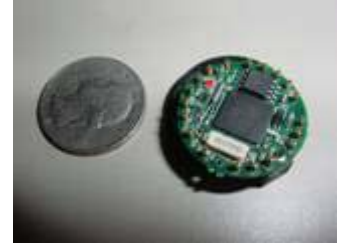
Effects of retransmission on transmission time
(128 packets per node)



Multiple Channels

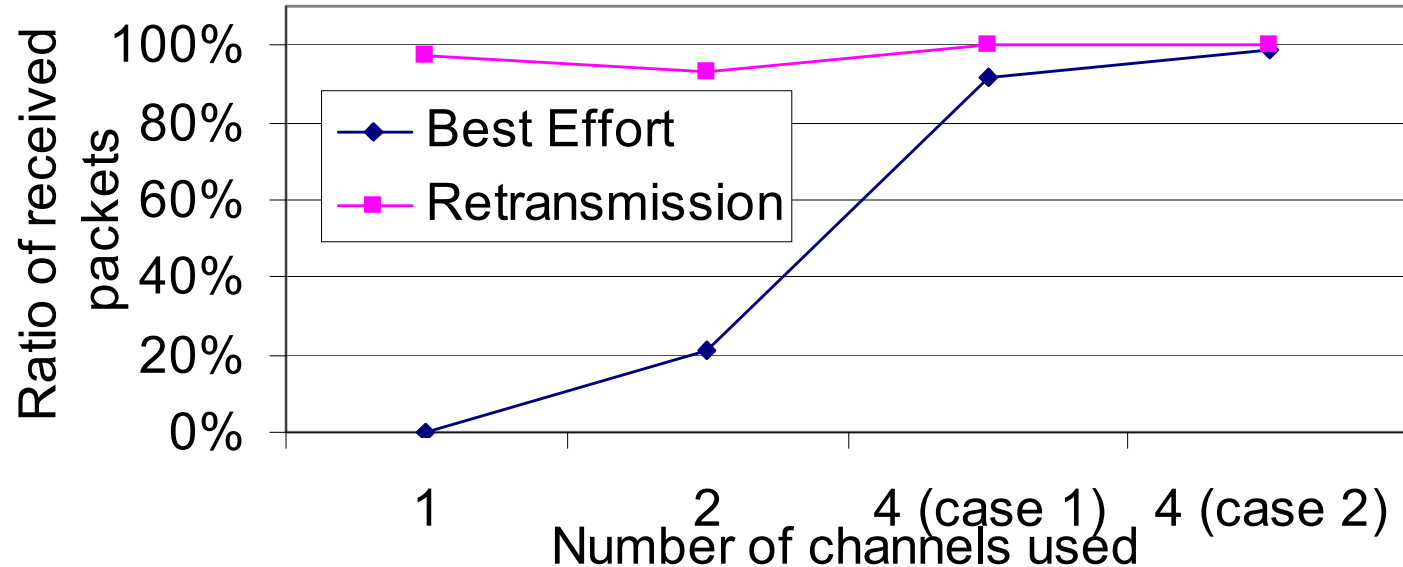


Cases with multiple channels

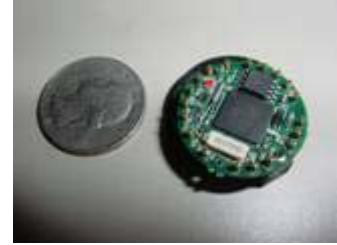


- **Using multiple channels reduced the packet losses due to collision.**

The effects of multiple channels on collision
(128 packets per node)

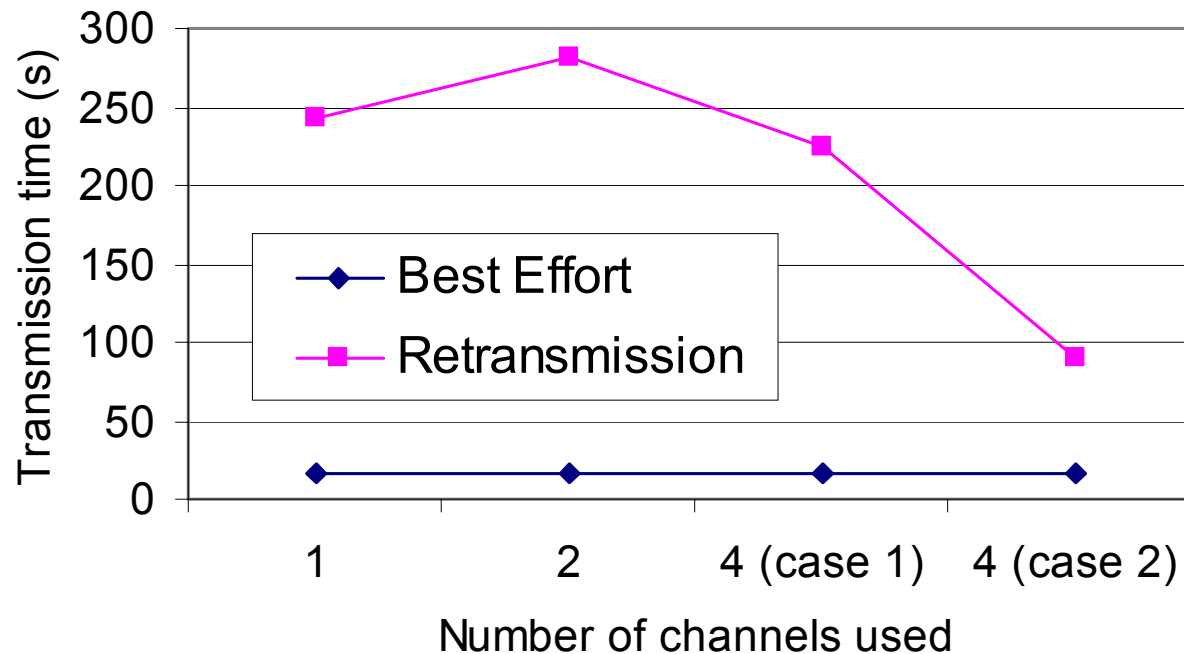


Cases with multiple channels

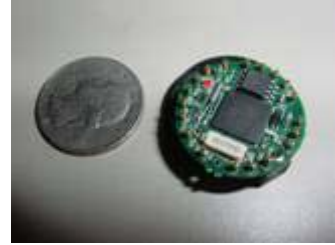


- Using multiple channels reduced the time cost to achieve high receiving rate

The effects of multiple channels on transmission time (128 packets per node)

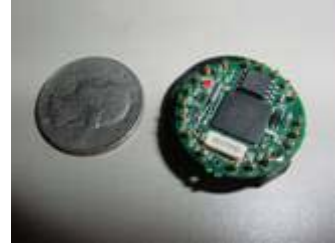


Discussion & Future Works



- **Comparison with MICA**
 - **Pros: Better coverage and reliability**
 - **Cons: Slower transmission (60 sec vs. 9 sec for 512 packets) caused by**
 - » **Slower clock rate of radio (19Kbps vs. 40Kbps)**
 - » **Less efficient interrupt handler**
 - **Modifying interrupt handler (from SPI to timer interrupt) will address this.**

Discussion & Future Works

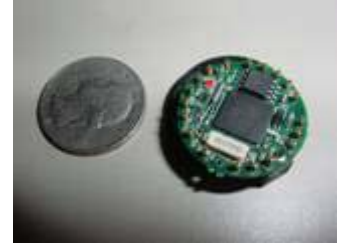


- **Problems with our reliable transmission method**
 - Effective for moderate collision, but not for high collision.
 - Introducing exponential back-off is expected to be helpful.
 - Overhead of retransmission is negligible.

Best Effort	Retransmission (5 retransmission)	Retransmission (0 retransmission)
31 sec	64 sec	32 sec

Time to send/receive 512 packets

Discussion & Future Works



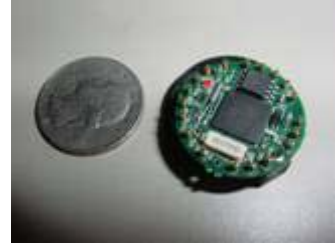
- **Using multiple channels**
 - Reduces collision.
 - Currently statically determined, vulnerable to misconfiguration.
 - Dynamic frequency allocation is needed.
- **Coding with error correction code**
 - The theoretical lower bound of code word is 13-bits without considering preamble and start symbol.
 - Existing implementation used 3 byte code word.
 - Reducing the code word to 2 bytes will be helpful.

End

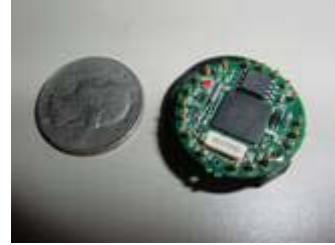


-
- **Questions?**

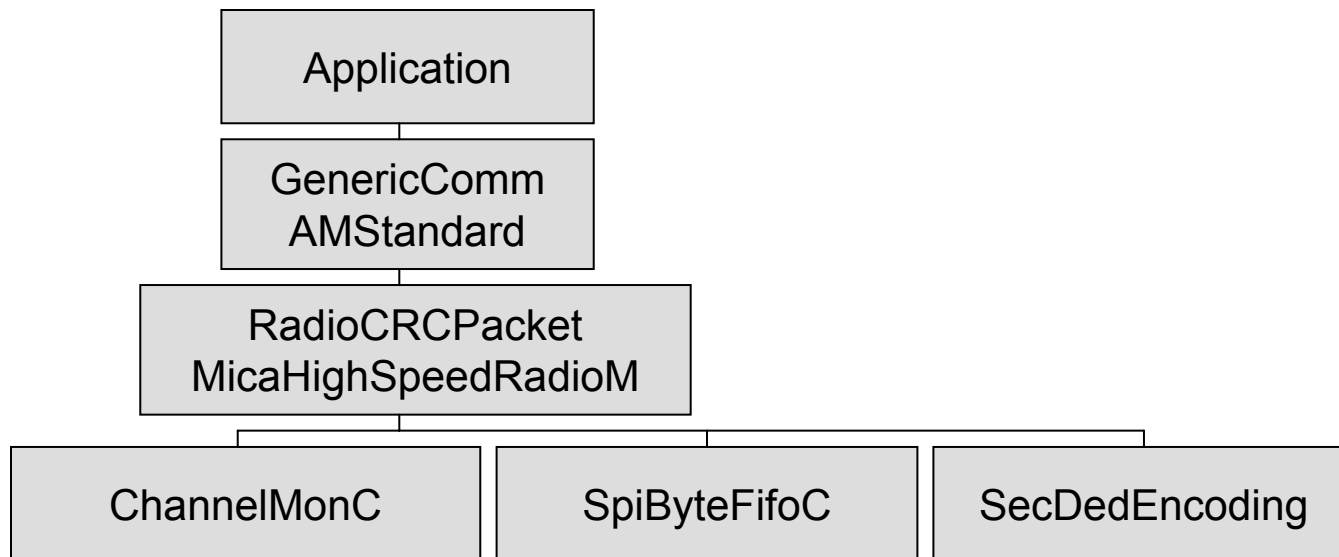
Extra Slides



Overview of existing network stack (MICA)



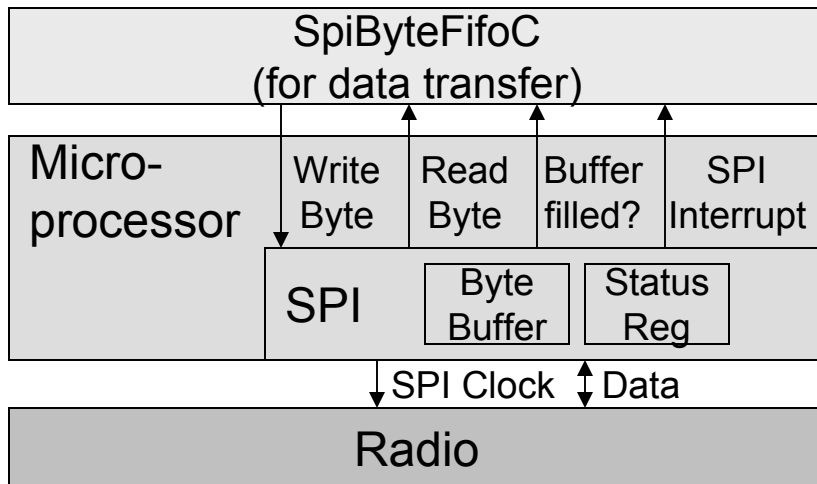
- **Converts a packet to and from raw bytes**
- **Sends and receives bytes**
- **Calculates CRC for sanity check**
- **Codes data with ECC**



Data interface to the radio

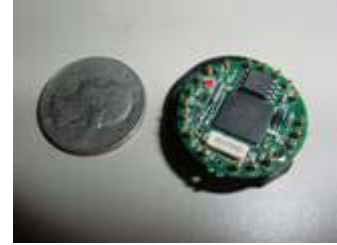


- **Microprocessor transfers data to and from the radio using byte level interface called SPI.**
- **SPI consists of byte buffer, status register and clock.**
- **At each clock interrupt, status register is checked for a received byte.**

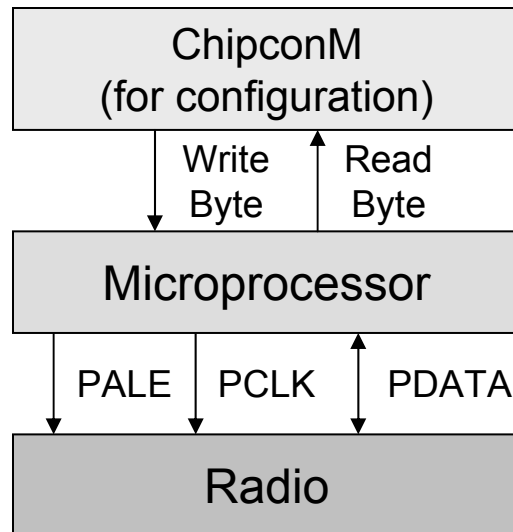


- **With no incoming byte, the microprocessor can send a byte into the byte buffer by setting the data direction as send.**

Configuring Chipcon Radio

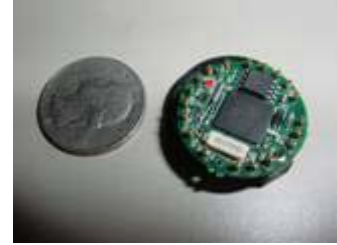


- The microprocessor needs to communicate with CC1000 radio chip to configure or monitor the status of it.
- The properties like operating frequency and power consumption can be set up by changing the CC1000 status registers.



- By setting or clearing three pins, the microprocessor can send or read a byte to a CC1000 status register.

Rayleigh Fading



- The graphs in outdoor tests consistently had dips at 900 ft.
- Radio waves from the sender can take different paths and cancel each other when the waves are of opposite phase.
- This is called Rayleigh Fading.

