An Energy-efficient MAC protocol for Wireless Sensor Networks

Wei Ye, John Heidemann, Deborah Estrin

presented by

Xiang Li

Outline

- Introduction to WSN
- Design considerations
- Energy inefficiency Sources
- Existing MAC protocol design
- S-MAC
- Testbed and Implementation
- Result and Analysis
- Conclusion and future work

Introduction to WSN

- Applications:
  - Nodes cooperate for a common task
  - In-network data processing
- Differences between WSN and ad-hoc network
  - Battery powered nodes → Energy efficiency
  - Large quantity of densely deployed nodes
  - This dense deployment brings high degree of interactions
  - Resources constraint
  - Auto configuration and auto organization

Design Considerations

- Level 1 issues
  - Collision avoidance-a basic task of MAC protocols
  - Good scalability
  - Energy efficiency

    -Often difficult recharge batteries or replace them

    -Prolonging the life-time is important
- Level 2 issues
  - Latency, fairness, throughput, bandwidth

Energy Inefficiency Sources

- Collision
  - Corrupted packets must be retransmitted and it increases energy consumption
- Overhearing
  - Receive packets destined to others

Energy Inefficiency sources

- Control packet overhead
- Idle listening
    - Listening to receive possible traffic that is not sent→ Dominant energy inefficiency factor in WSN
    - Consumes 50-100% of the energy required for receiving

Existing MAC protocol design

- Contention based protocols
  - IEEE 802.11 distributed coordination function (DCF)
    - Each nodes contends for the medium as necessary and wastes a lot of energy in idle listening
  - PAMAS
    - Asking separate radio channel for RTS/CTS
    - Does not address the issue of reduce idle listening

- TDMA based protocols
  - Advantages
    - lower energy conservation when compared to contention based as the duty cycle of the radio is reduced and no contention overhead
  - Problems
    - Requires nodes to form real communication clusters and managing inter-cluster communication is difficult
    - It is not easy to change the slot assignment dynamically, hence scalability is not as good as contention based

S-MAC

- Design
  - Goal
    - Reduce energy consumption
    - Support good scalability and collision avoidance
  - Solutions to energy inefficiency issues
    - Collision avoidance - using RTS and CTS
    - Overhearing avoidance - switching the radio off when the transmission is not meant for that node
    - Control overhead - Message Passing
    - Idle listening - Periodic listen and sleep

Trade offs between energy efficiency and performance (1)

- Trade-offs of per-hop fairness and latency
- This does not necessarily result in lower end-to-end fairness and latency

Trade offs between energy efficiency and performance (2)

- It is important in wireless voice or data networks as each user desires equal opportunity and time to access the network
- Is it important for sensor networks?
  - In sensor networks all nodes co-operate and work together for a single application
  - So per-hop fairness is not important as long as application level performance is not degraded.

Network assumptions (1)

- Composed of many small nodes deployed in an ad hoc fashion
- Most communication will be between nodes as peers, rather than to a single base station
- Nodes must self-configure

Network assumptions (2)

- Dedicated to a single application or a few collaborative applications
- Involves in-network processing to reduce traffic and thereby increase the life-time
- This implies that data will be processed as whole messages at a time in store-and-forward fashion
- Hence packet or fragment-level interleaving from multiple sources only delays overall latency
- Applications will have long idle periods and can tolerate some latency

Periodic listen and Sleep (1)

- Problem: Idle listing wastes a lot of energy
- Solution: Periodic listen and sleep

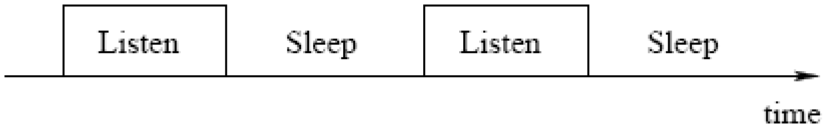

Fig. 1. Periodic listen and sleep.

- Turn off radio when sleeping
- Reduce duty cycle to ~ 10% (200ms on/2s off)

Periodic listen and Sleep (2)

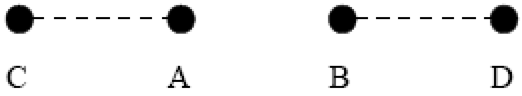

Fig. 2. Neighboring nodes A and B have different schedules. They synchronize with nodes C and D respectively.

- Not all neighboring nodes can synchronize together
- Two neighboring nodes (A and B) can have different schedules if they are required to synchronize with different node

Periodic listen and Sleep (3)

- If a node A wants to talk to node B, it just waits until B is listening
- If multiple neighbors want to talk to a node, they need to contend for the medium
- Contention mechanism is the same as that in IEEE802.11 (using RTS and CTS)
- After they start data transmission, they do not go to periodic sleep until they finish transmission

Periodic listen and Sleep (4)- choosing and maintaining schedules

- Each node maintains a schedule table that stores schedules of all its known neighbors.
- To establish the initial schedule (at the startup) following steps are followed:
  - A node first listens for a certain amount of time.
  - If it does not hear a schedule from another node, it randomly chooses a schedule and broadcast its schedule immediately.
  - This node is called a SYNCHRONIZER.
  - If a node receives a schedule from a neighbor before choosing its own schedule, it just follows this neighbor's schedule.
  - This node is called a FOLLOWER and it waits for a random delay and broadcasts its schedule.
  - If a node receives a neighbor's schedule after it selects its own schedule, it adopts to both schedules and broadcasts its own schedule before going to sleep.

Periodic listen and Sleep (5)- Maintaining Synchronization

- Timer synchronization among neighbors are needed to prevent the clock drift.
- Done by periodic updating using a SYNC packet.
- Updating period can be quite long as we don’t require tight synchronization.
- Synchronizer needs to periodically send SYNC to its followers.
- If a follower has a neighbor that has a different schedule with it, it also needs update that neighbor.

- Time of next sleep is relative to the moment that the sender finishes transmitting the SYNC packet
- Receivers will adjust their timer counters immediately after they receive the SYNC packet
- Listen interval is divided into two parts: one for receiving SYNC and other for receiving RTS

Periodic listen and Sleep (6)- Timing Relationship of Possible Situations

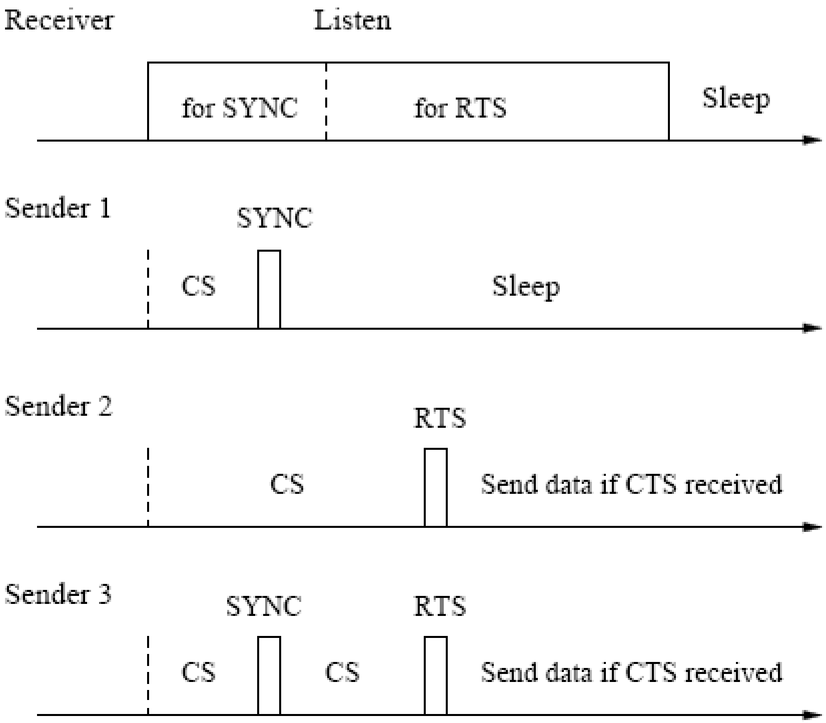

Fig. 3. Timing relationship between a receiver and different senders. CS stands for carrier sense.

Collision Avoidance

- Problem: Multiple senders want to talk
- Solution: Following IEEE 802.11 ad hoc procedures
  - Physical and virtual carrier sense
  - Randomized backoff time
  - RTS/CTS for hidden terminal problem
  - RTS/CTS/DATA/ACK sequence

Overhearing Avoidance

- Problem: Receive packets destined to others
  - In 802.11, each node keeps listening to all transmissions from its neighbors for virtual carrier sensing
  - Each node should overhear a lot of packets not destined to itself
- Solution: Letting interfering nodes go sleep after they hear an RTS or CTS packet
- Which nodes should sleep?
  - All immediate neighbors of sender and receiver
  - S-MAC lets interfering nodes go to sleep after they hear an RTS or CTS
    - DATA packets are normally much longer than control packets
- How long?
  - The *duration* field in each packet informs other nodes the sleep interval
  - After hearing the RTS/CTS packet destined to a node, all the other immediate neighbors of both the sender and receiver should sleep until the NAV becomes zero

Message Passing

- Problem: Sensor net in-network processing requires *entire* message
- Solution: Don't interleave different messages
    - Long message is fragmented & sent in burst
    - RTS/CTS reserve medium for entire message
    - Fragment-level error recovery ─ ACK

      ─ Extend Tx time and re-transmit immediately if no ACK is received
- Advantages
    - Reduces latency of the message
    - Reduces control overhead
- Disadvantage
    - Node-to-node fairness is reduced, as nodes with small packets to send has to wait till the message burst is transmitted

Protocol Implementation

- Testbed
  - Rene motes, developed at UCB
  - They run TinyOS, an event-driven operating systems
  - Two type of packets
    - Fixed size data packets with header (6B), payload (30B) and CRC (2B)
    - Control packets (RTS and CTS), 6B header and 2B CRC

MAC modules implemented

- Simplified IEEE 802.11 DCF – physical and virtual carrier sense, backoff and retry, RTS/CTS/DATA/ACK packet exchange and fragmentation support
- Message passing with overhearing avoidance
- The complete S-MAC – all the features are implemented

Topology

- Two-hop network with two sources and two sinks
- Sources generate message which is divided into fragments
- Traffic load is changed by varying the inter-arrival period of the message

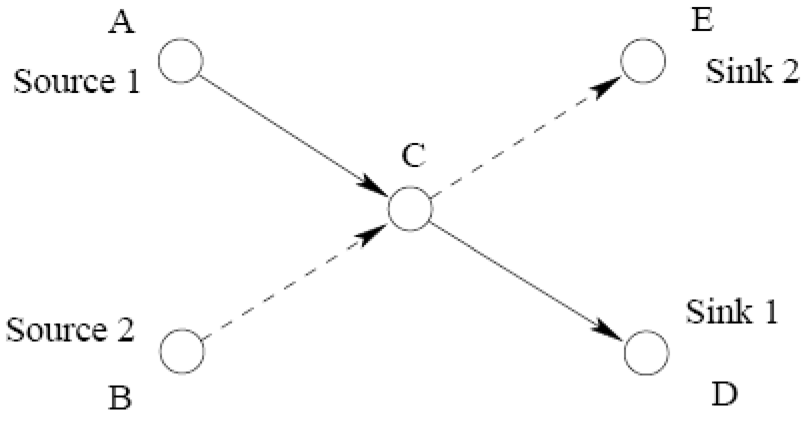

Fig. 7. Topology used in experiments: two-hop network with two sources and two sinks.

Energy consumption in the source nodes

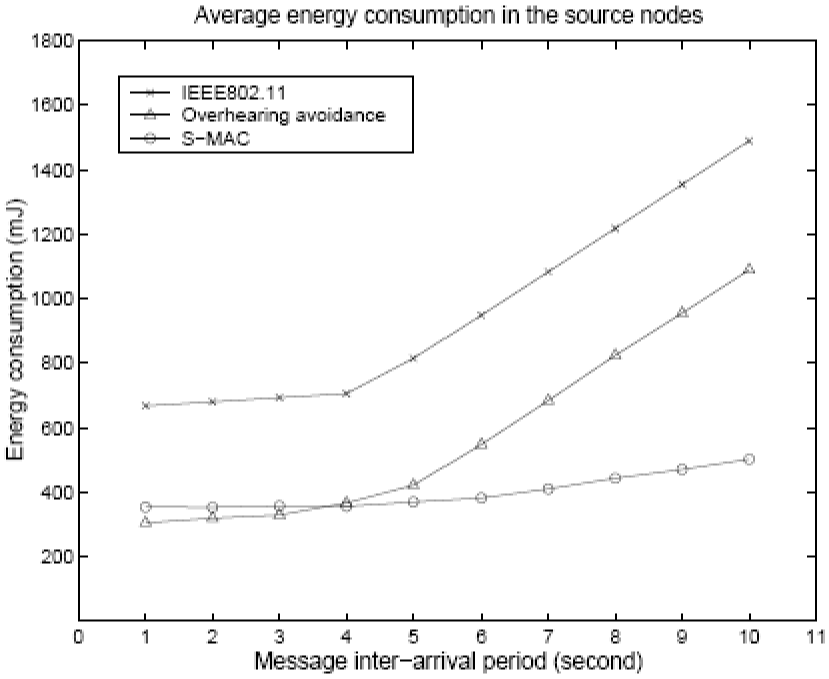
Average energy consumption in the source nodes

Fig. 8. Measured energy consumption in the source nodes.

Percentage of time that the source nodes are in the sleep mode

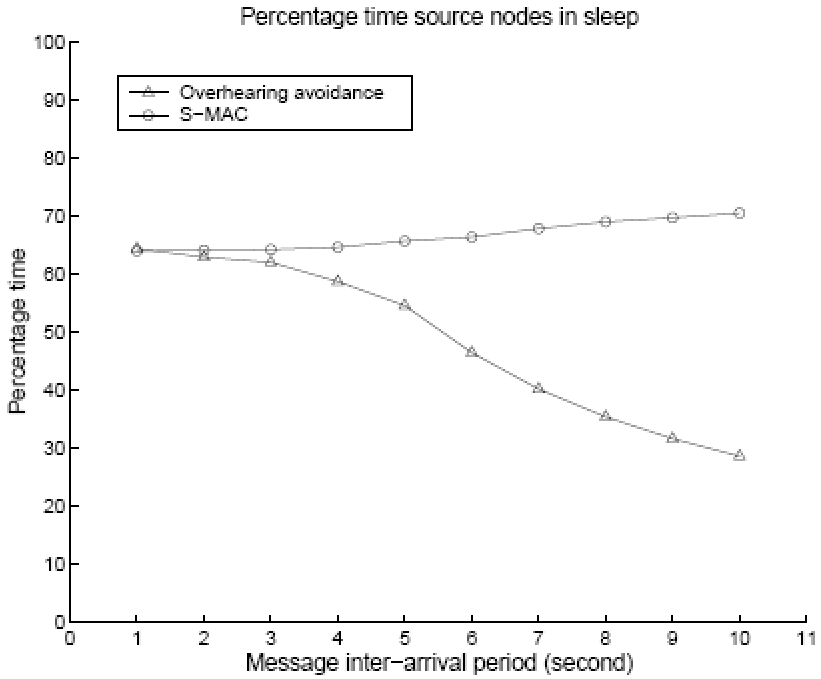

Fig. 9. Measured percentage of time that the source nodes in the sleep mode.

Energy consumption in the intermediate node

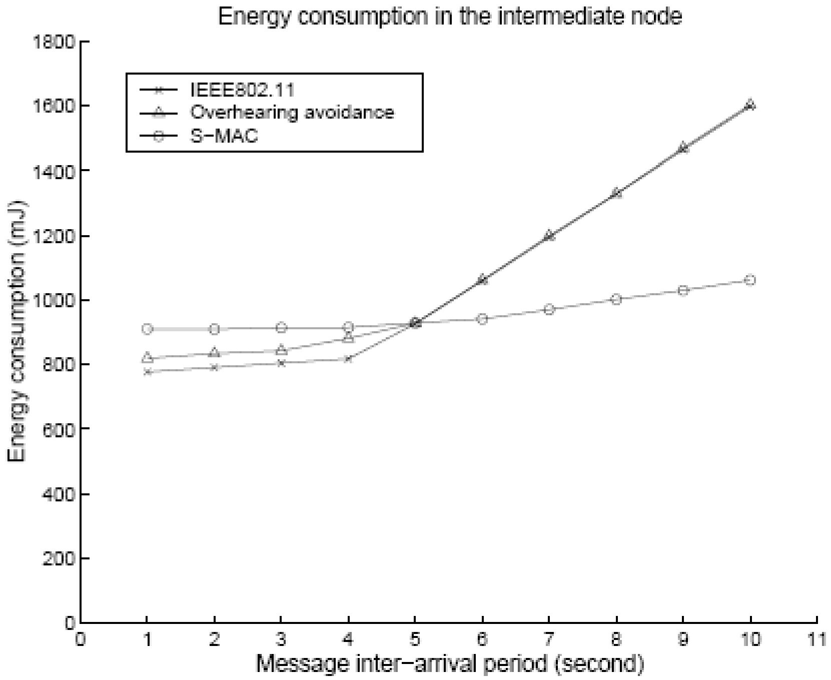

Fig. 10. Measured energy consumption in the intermediate node.

Conclusions and Future work

- Conclusion:
  - S-MAC offers a significant improvement on energy efficiency properties comparing to IEEE 802.11
- Future work
  - Experiment on large scale deployment
  - Analysis on the effects brought by topology change