Ultra-Low Duty Cycle MAC with Scheduled Channel Polling

(Wei Ye, Fabio Sliva, and John Heidemann)

Advanced Computer Networks

ECE537-2014 Fall

Presented by Tianyang Wang

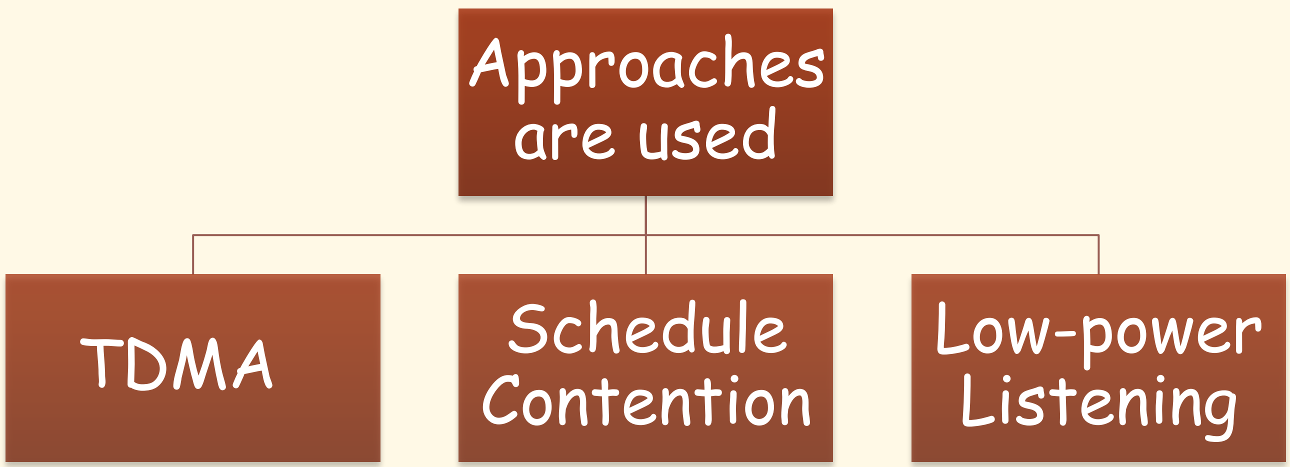
Dec.2, 2014

Outline

1. Introduction
2. Design of SCP-MAC
3. Performance
4. Protocol Implementation
5. Experimental Evaluation
6. Conclusion

Introduction

Major sources of energy waste are idle listening, collision, overhearing and control overhead. And idle listening is a dominant factor in most sensor network applications.



Introduction

Schedule: The schedule determines when a node should listen and when it should sleep

Scheduling reduces energy cost by ensuring that listeners and transmitters have a regular, short period in which to exchange data and can sleep at other times.

Low-power Listening

Low-power Listening (LPL): In LPL, nodes wake up very briefly to check channel activity without actually receiving data. According to the paper, we call this action channel polling. If the channel is idle, the node immediately goes back to sleep. Otherwise it stay awake to receive data. Although nodes regularly poll the channel with a pre-defined polling period, their polling times are not explicitly synchronized. To connect with receivers, senders send a long preamble before each message, which is guaranteed to intersect with a polling.

Low-power Listening

Three major problems: 1. receiver and polling efficiency is gained at the much greater cost of senders. 2. the balance between sender and receiver costs makes LPL-based protocols very sensitive to tuning for an expected neighborhood size and traffic rate. 3. it is challenging to adapt LPL directly to newer radio like 802.15.4, since the specification limits the preamble size.

Outline

1. Introduction
2. **Design of SCP-MAC**
3. Performance
4. Protocol Implementation
5. Experimental Evaluation
6. Related Work and Conclusion

Main goals

1. To push the duty cycle an order of magnitude lower than is practical with current MAC protocols.

2. To adopt to variable traffic loads common in many sensor network applications

Synchronized Channel Polling

LPL: Nodes poll channel asynchronously to test for possible traffic. To send a packet, the sender adds a preamble before the packet. This preamble is effectively a wake-up signal, informing other nodes. The preamble must be at least as long as the channel polling period to ensure all receivers will detect it. The performance of LPL is sensitive to the channel polling period, since longer periods reduce receiver costs but increase sender costs. Selecting an optimal value requires knowledge of network size and completely periodic traffic.

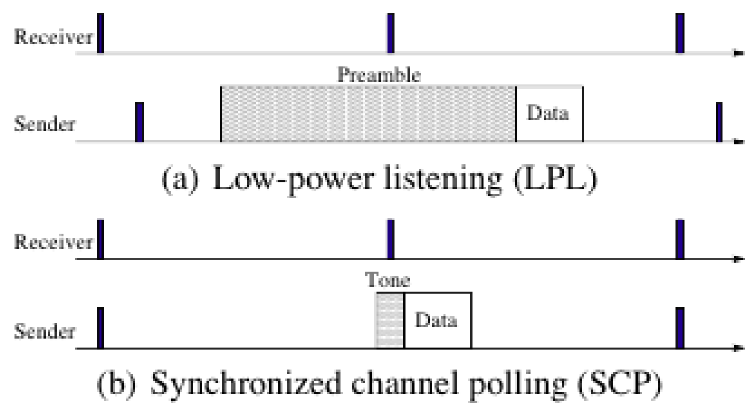


Figure 1. Sender and receiver synchronization schemes.

Synchronized Channel Polling

SCP-MAC adopts channel polling from LPL approaches.
SCP-MAC synchronizes the polling times of all neighboring nodes.
The primary advantage of scheduled polling is that only a short wake-up tone is required for senders to guarantee the connection.
2. Synchronization reduces the cost of overhearing, since on average all nodes will hear half the preamble before waking up, even for packets addressed to other receivers.
3. With synchronization SCP works efficiently for both unicast and broadcast traffic, while some existing optimizations to improve LPL work only for unicast.
4. Short wakeup tones make SCP-MAC more robust to varying traffic load.
The penalty of scheduled polling is the cost of maintaining schedule synchronization, and potentially the requirement of maintaining multiple schedules.
Each node broadcasts its schedule in a SYNC packet to its neighbors every synchronization period.

Adaptive Channel Polling

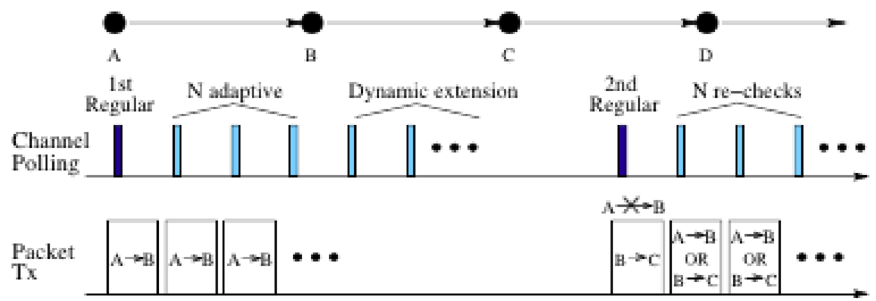


Figure 2. Adaptive channel polling and multi-hop streaming.

Consider that node A sends to node B. When B receives a packet during the first regular polling, it adds n high-frequency polls in the same frame, immediately following its regular poll. If A has more packets to send, it sends them in these adaptive polling times. Spacing of adaptive slots is determined by the longest packet length that physical layer supports.

Multi-hop Streaming

To quickly bring all nodes on the path into the high-rate polling mode, and keep them in this mode until the burst ends. In this way, data can be quickly streamed from the source to the sink.

In order to shift node C quickly to adaptive polling, node A intentionally gives up the transmission opportunity in the second regular polling slot, allowing B to send to C without contention from A. When node C receives this packet, it too will shift to adaptive polling. The same procedure repeats.

Multi-hop Streaming

How many polls should be added?

In a multi-hop network, each node contents with its previous- and next-hop nodes if they all have data to send. Thus, each of the three nodes needs a slot to send, so that packets can quickly proceed downstream. Therefore, the authors set the number of adaptive polling slots n to 3.

Other Optimizations

Two-Phase Contention: Before sending a tone, a node performs carrier sense by randomly select a slot within the first contention window. Only idle channel allows the node to proceed.
If a node successfully send wakeup tones, it will enter the second contention window. If such a node still detects channel idle in the second contention phase, it starts sending data.

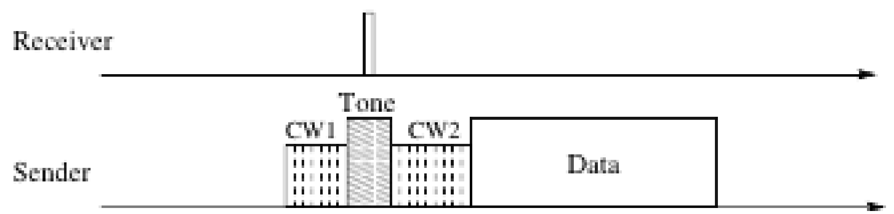


Figure 3. Two-phase contention in SCP-MAC.

The two-phase contention lowers the collision probability. So it can decrease the energy cost because of collision.

Other Optimization

Overhearing Avoidance Based on Headers:
When RTS/CTS is disabled, a receiver examines the destination address of a packet immediately after receiving its MAC header, before completely receiving the packet. If it is a unicast packet destined to another node, it immediately stops the reception and places the radio to sleep.

Outline

1. Introduction
2. Design of SCP-MAC
3. **Performance**
4. Protocol Implementation
5. Experimental Evaluation
6. Conclusion

Models and Metrics

$$
\begin{aligned}
E &= E_{cs} + E_{tx} + E_{rx} + E_{poll} + E_{sleep} \\
&= P_{listen}t_{cs} + P_{tx}t_{tx} + P_{rx}t_{rx} \\
&\quad + P_{poll}t_{poll} + P_{sleep}t_{sleep} \qquad (1)
\end{aligned}
$$

The whole energy cost equals the energy cost by carrier sense, transmission, receive, channel polling and sleep.

Asynchronous Channel Polling: LPL

According to the function 1, we need t_{cs}, t_{tx}, t_{rx}, t_{poll}, t_{sleep}

$$L_{preamble} = T_p / t_B \qquad (2)$$

$$t_{cs} = t_{cs1} / T_{data} = t_{cs1} r_{data} \qquad (3)$$

$$\begin{aligned} t_{tx} &= (L_{preamble} + L_{data}) t_B r_{data} \\ &= (T_p + L_{data} t_B) r_{data} \qquad (4) \end{aligned}$$

$$t_{rx} = n(T_p/2 + L_{data} t_B) r_{data} \qquad (5)$$

$$t_{poll} = t_{p1} / T_p \qquad (6)$$

$$t_{sleep} = 1 - t_{cs} - t_{tx} - t_{rx} - t_{poll} \qquad (7)$$

Asynchronous Channel Polling: LPL

$$
\begin{aligned}
E_r \quad = \quad & (P_{listen} t_{cs1} + P_{tx}(T_p + t_{pkt}) + nP_{rx}(T_p/2 + t_{pkt}))r_{data} \\
& + P_{poll} t_{p1}/T_p \\
& + P_{sleep}(1 - (t_{cs1} + (n/2+1)T_p + (n+1)t_{pkt})r_{data} \\
& - t_{p1}/T_p) \qquad (8)
\end{aligned}
$$

WPI

Asynchronous Channel Polling: LPL

In order to get the limitation of the energy cost, the authors let the differential coefficient equal to zero.

$$T_{p,r}^{*} = \sqrt{\frac{(P_{poll} - P_{sleep})t_{p1}}{r_{data}(P_{tx} + nP_{rx}/2 - (n/2+1)P_{sleep})}} \qquad (10)$$

In this way, they get the optimal preamble length.

Polling Period and Wakeup-tone

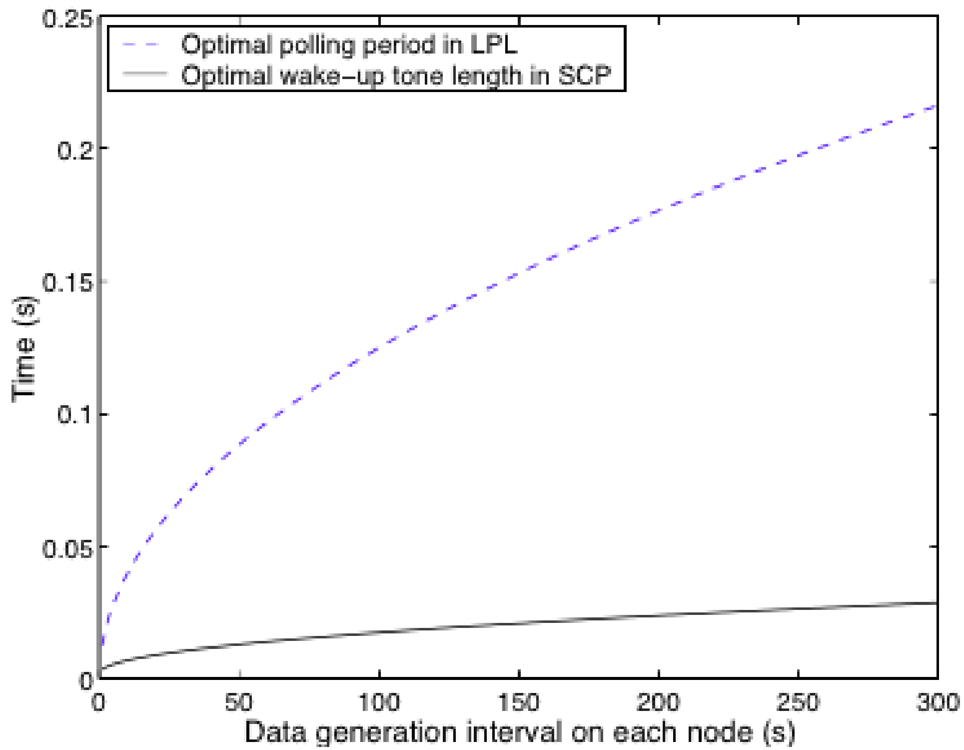


Figure 4. Optimal channel polling period in LPL (dotted), and wakeup-tone length in SCP (solid), given neighborhood size of 10.

Scheduled Channel Polling: SCP

Synchronization Requirement and Tradeoffs

$$t_{diff} = 2T_{sync} r_{clk} \qquad (11)$$

T_{sync} is synchronization period and r_{clk} is the clock drift. When we consider n+1 nodes, we can get the guard time and the duration of the wake-up tone.

$$t_{guard} = 4T_{sync} r_{clk} / (n+1) \qquad (12)$$

$$t_{tone} = 4T_{sync} r_{clk} / (n+1) + t_{mtone} \qquad (13)$$

Perfect Piggybacking

Given the fact that may types of data transmissions in sensor networks are periodic, synchronization information can be easily piggybacked on data.

$$
\begin{aligned}
E_{sp} = \; & P_{listen} t_{cs1} r_{data} \\
& + (P_{tx} + nP_{rx})(t_{tone} + L_{sB} t_B + L_{data} t_B) r_{data} \\
& + P_{poll} t_{p1} / T_p \\
& + P_{sleep} [1 - t_{cs1} r_{data} \\
& \quad - (n+1)(t_{tone} + L_{sB} t_B + L_{data} t_B) r_{data} \\
& \quad - t_{p1} / T_p] \qquad (16)
\end{aligned}
$$

Perfect Piggybacking

Ideally, with the periodic traffic from all neighbors, a node should only poll the channel when there is a transmission from a neighbor.

$$T^*_{p,sp} = \frac{1}{n(r_{data})} \tag{17}$$

All Explicit Synchronization

The authors consider the worst case, assuming no piggybacking is possible and so all synchronization must be done with messages dedicated for that purpose.

$$t_{cs} = t_{cs1}(r_{data} + r_{sync}) \tag{18}$$

$$t_{tx} = (t_{tone} + L_{data} t_B) r_{data} + (t_{tone} + L_{sync} t_B) r_{sync} \tag{19}$$

$$t_{rx} = n(t_{tone} + L_{data} t_B) r_{data} + n(t_{tone} + L_{data} t_B) r_{sync} \tag{20}$$

All Explicit Synchronization

$$\begin{aligned} E_{snp} = {} & P_{listen} t_{cs1} (r_{data} + r_{sync}) \\ & + (P_{tx} + nP_{rx})(t_{tone} + L_{data} t_B) r_{data} \\ & + (P_{tx} + nP_{rx})(t_{tone} + L_{sync} t_B) r_{sync} \\ & + P_{poll} t_{p1} / T_p \\ & + P_{sleep} [1 - t_{cs1} (r_{data} + r_{sync}) \\ & \quad - (n+1)(t_{tone} + L_{data} t_B) r_{data} \\ & \quad - (n+1)(t_{tone} + L_{sync} t_B) r_{sync} \\ & \quad - t_{p1} / T_p] \end{aligned}$$

Then, when we ignore the energy consumption in sleep state, we can get the optimal polling period for scheduled polling with independent SYNC packets

$$T^*_{p,snp} = \frac{1}{n(r_{data} + r_{sync})} \qquad (22)$$

All Explicit Synchronization

What is the optimal synchronization period that minimizes E_{snp}?

We can also get T_{sync} through finding the differential coefficient.

$$T^*_{sync} = \sqrt{\frac{n(n+1)(E_l + P_t t_t + E_p)}{2 r_{data} r_{clk} P_t}} \qquad (24)$$

Once T^*_{sync} is known, we can obtain the optimal tone duration.

$$t^*_{tone} = \frac{4 T^*_{sync} r_{clk}}{n+1} + t_{mtone} \qquad (25)$$

Optimal SYNC period for SCP-MAC

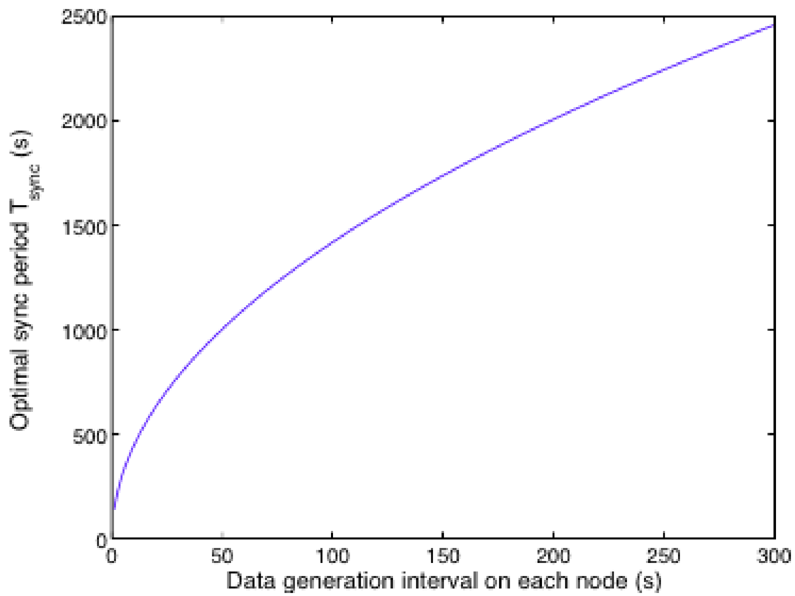


Figure 5. Optimal SYNC period for SCP-MAC.

First, this observation suggests that synchronization overhead can be low.
Second, clock synchronization and scheduled polling allows much shorter preambles
Third, when piggybacking is used, synchronization happens for free on top of data.

Performance

1. LPL consumes about 3-6 times more energy than SCP on the CC1000, because of the expense of long preamble.
2. Piggybacking reduces energy by half when data is sent rarely; the benefits are minimal when data is sent frequently because the cost of data packets then overwhelm control costs.
3. Because the preamble length still must be at least the length of the polling period, regardless of the radio speed, the cost of LPL increase. The energy cost of SCP falls, because it takes shorter time to send data and perform carrier sense on the high-speed radio.

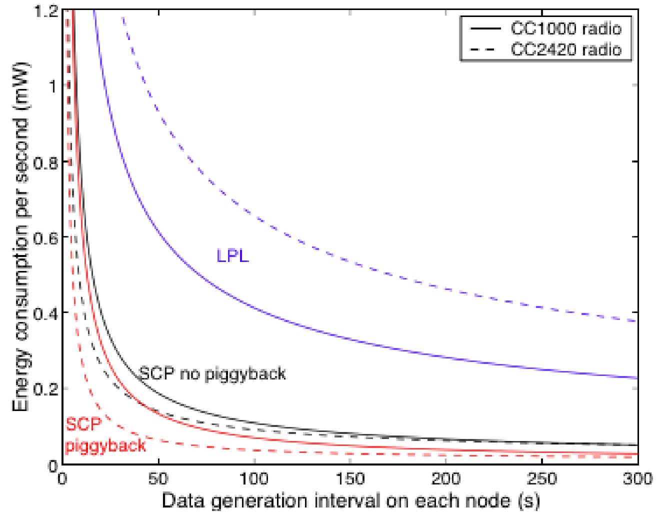


Figure 6. Analysis of optimal energy consumption for LPL and SCP with and without piggyback for CC1000 (solid lines) and CC2420 (dashed).

Outline

1. Introduction
2. Design of SCP-MAC
3. Performance
4. **Protocol Implementation**
5. Experimental Evaluation
6. Conclusion

Protocol Implementation

The authors describe the software architecture of SCP-MAC in TinyOS.

- **PHY**: It handles the radio states
- **CSMA**: Provide a common service to both LPL and SCP
- **LPL**: Its major purpose is to periodically poll the channel and send radio to sleep when there is no activity
- **Scheduling**: It uses basic LPL to bootstrap schedules with SYNC packets

Differences of CC2420 and CC1000

First, CC2420 is a packet-level radio, and the microcontroller cannot get byte-level access.
This potentially affects the accuracy of time synchronization.
Second, the radio automatically generates a preamble for each packet to comply with 802.15.4 standard.
It limits the preamble length to 16 bytes with a fault length of 4 bytes.
This is a strong challenge to implement the long LPL preambles, and also forces SCP to use a normal packet as the wakeup tone.

Outline

1. Introduction
2. Design of SCP-MAC
3. Performance
4. Protocol Implementation
5. **Experimental Evaluation**
6. Conclusion

Experimental Evaluation

The energy consumption for SCP is almost constant at all rates, as the cost of sending each packet is about same. With broadcast traffic, all explicit SYNC packets are suppressed due to piggybacking.

For LPL, the energy consumption increases at lower rates, since the optimal polling interval is longer, therefore the cost on longer preamble is larger.

SCP can save more energy because scheduling allows a much shorter wakeup tone on each data message.

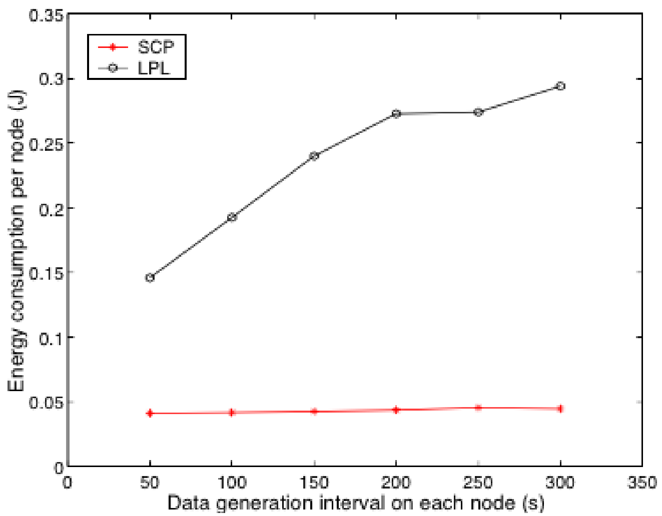


Figure 8. Mean energy consumption (J) for each node as traffic rate varies (assuming optimal configuration and periodic traffic).

Experimental Evaluation

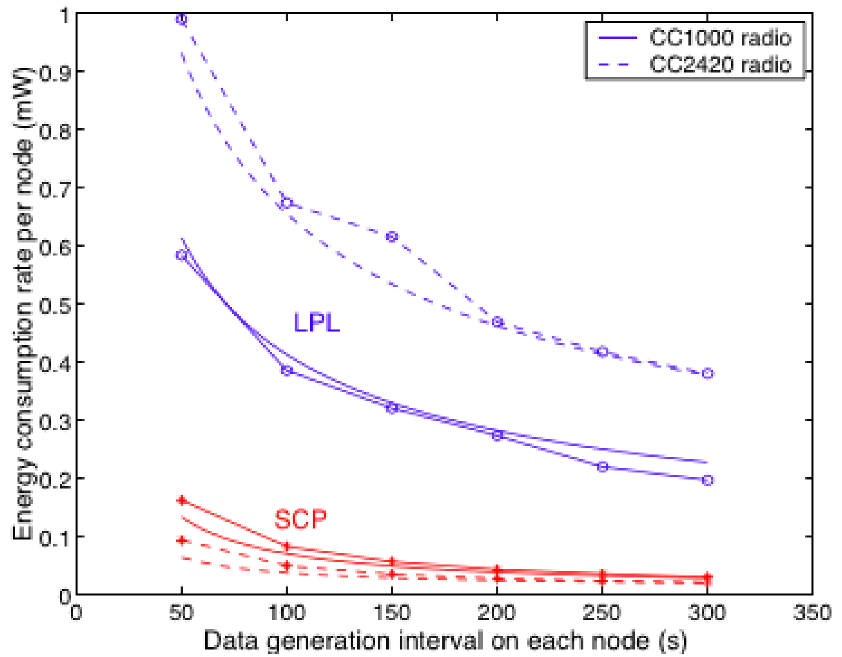


Figure 9. Mean energy consumption rate (J/s or W) for each node as traffic rate varies (assuming optimal configuration and periodic traffic). The radios are the CC1000 (solid lines) and CC2420 (dashed).

The experimental results confirm that the energy consumption of LPL increases on the faster radio, while that of SCP decrease.

Performance with Unanticipated Traffic

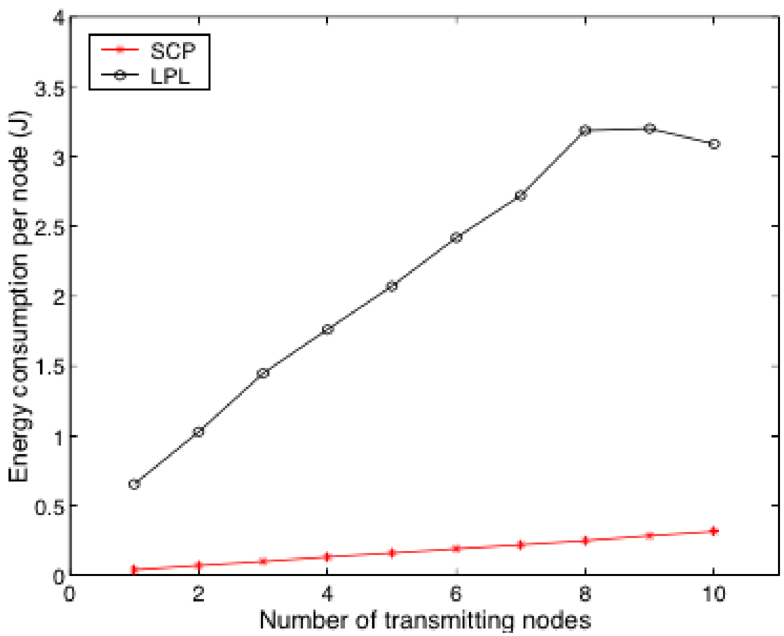


Figure 10. Energy consumptions on heavy traffic load with very low duty cycle configurations.

Disable adaptive polling and overhearing avoidance in SCP-MAC

LPL consumes about 8 times more energy than SCP to transmit an equal amount of data

The main reason is the high cost of LPL preamble.

Performance with Unanticipated Traffic

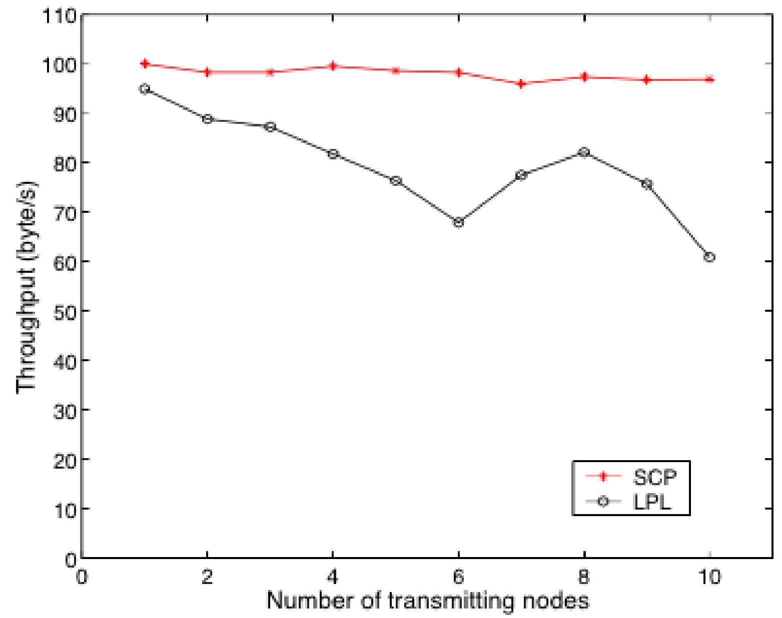


Figure 11. Throughput on heavy traffic load with very low duty cycle configurations.

This figure shows the throughput on heavy traffic load. It proves that two-phase contention which can decrease the collision has a very good performance.

Performance in a Multi-hop Network

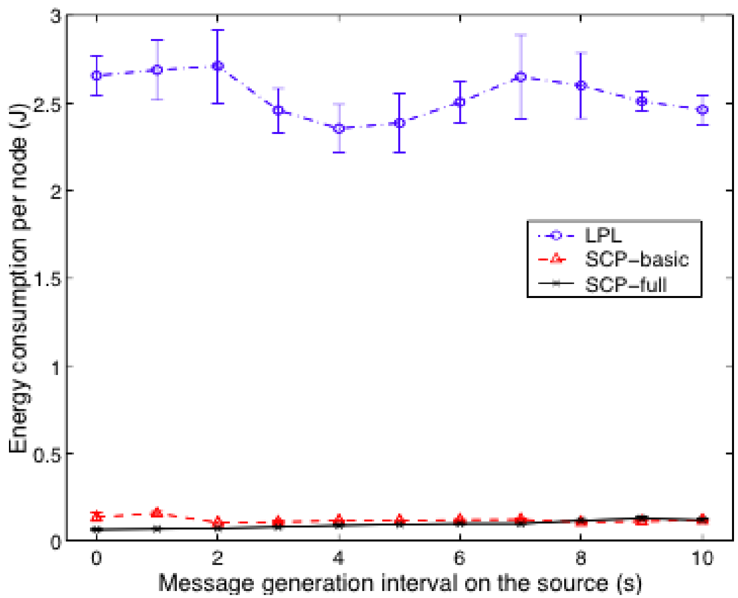


Figure 12. Mean energy consumption per node for multi-hop experiments (20 packets over 9 hops).

LPL's long preambles, both on reception of packets at each hop, and also due to reception by overhearers.

Performance in a Multi-hop Network

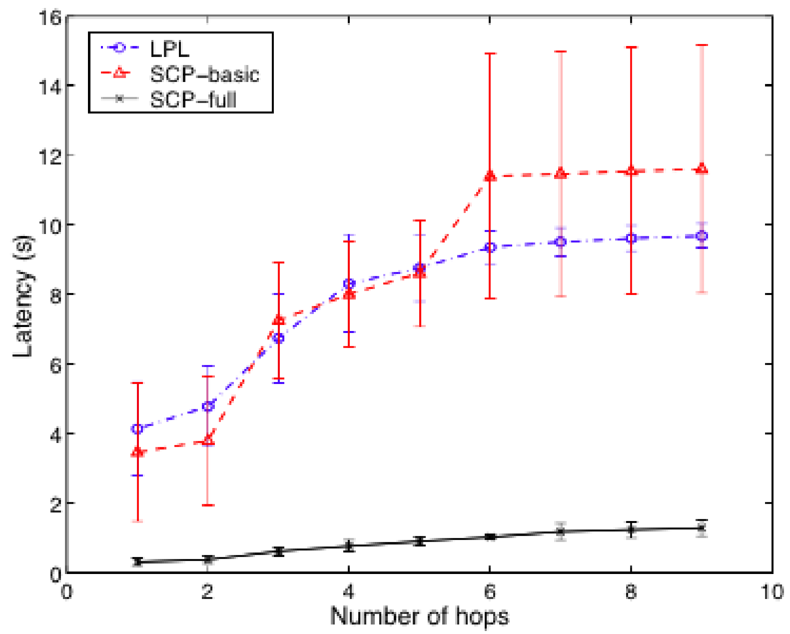


Figure 13. Mean packet latency over 9 hops at the heaviest load.

Adaptive polling not only enables adjacent nodes to send multiple packets in one polling interval, it also enables multi-hop streaming.

Outline

1. Introduction
2. Design of SCP-MAC
3. Performance
4. Protocol Implementation
5. Experimental Evaluation
6. **Conclusion**

Conclusions

1. By optimally combining scheduling and channel polling, SCP can operate sensor networks at duty cycles of 0.1% and lower.
2. SCP-MAC can robustly handle bursty and varying traffic loads, and adaptive channel polling significantly reduces latency by enabling multi-hop streaming.
3. The relative performance of SCP improves on newer, faster radios like the CC2420 while that of LPL degrades.