CSE 4215/5431:
Mobile Communications

Winter 2013

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cse.yorku.ca/course/4215

Some slides are adapted from the Schiller book website

Some proposed protocols

- All demand assignment protocols
- All two phase protocols
  - Reservation phase (minislots)
  - Data transmission phase (slots)
- Differ in how reservation phase works

Protocol 1 (RMAV)

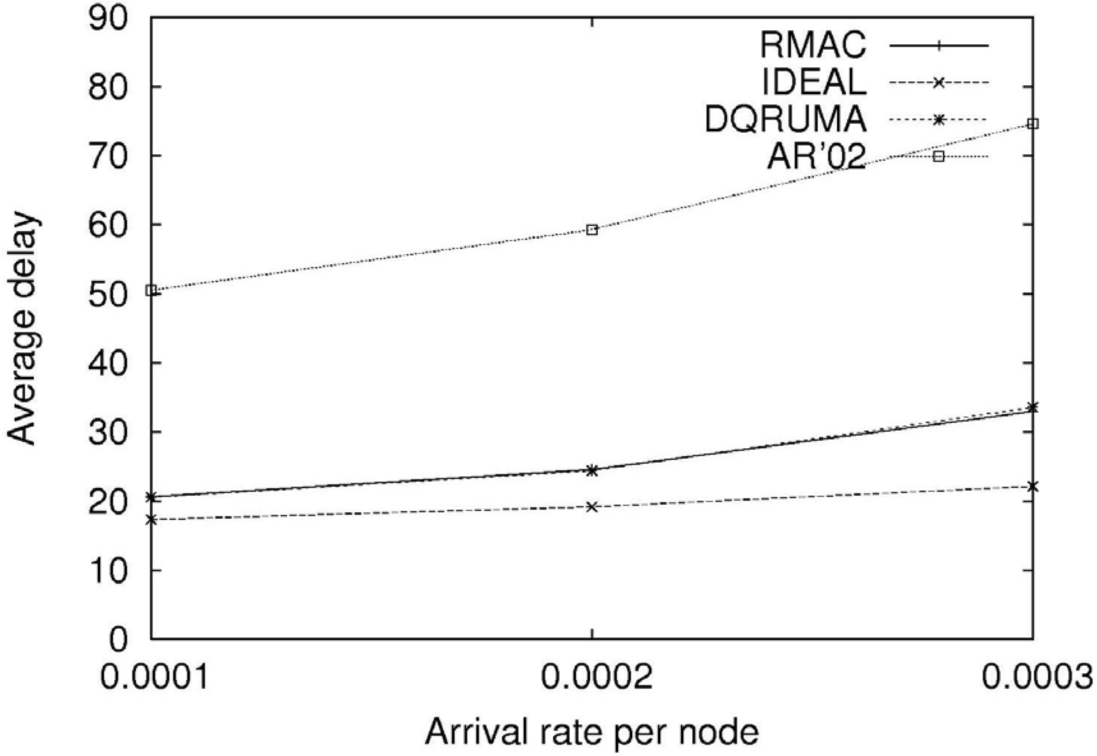
- D. Jeong, C. Hoi, and W. Jeon, "Design and performance evaluation of a new medium access control protocol for local wireless data communications," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 742–752, 1995.
- 1 reservation minislot every round
- Adaptive change of backoff probability
  - If collision, p doubled
  - If idle p halved
  - Else p unchanged
- Advantages and Disadvantages?

Protocol 2

- M. Abolelaze and R. Radhakrishnan, "The performance of a new wireless MAC protocol," in *3G Wireless 2002*.
- Adapt number of minislots each reservation round
  - if the number of minislots with collisions is larger than the number of empty minislots, then the number of minislots are doubled in the next round.
  - Else halve the number of minislots
- Advantages and Disadvantages?

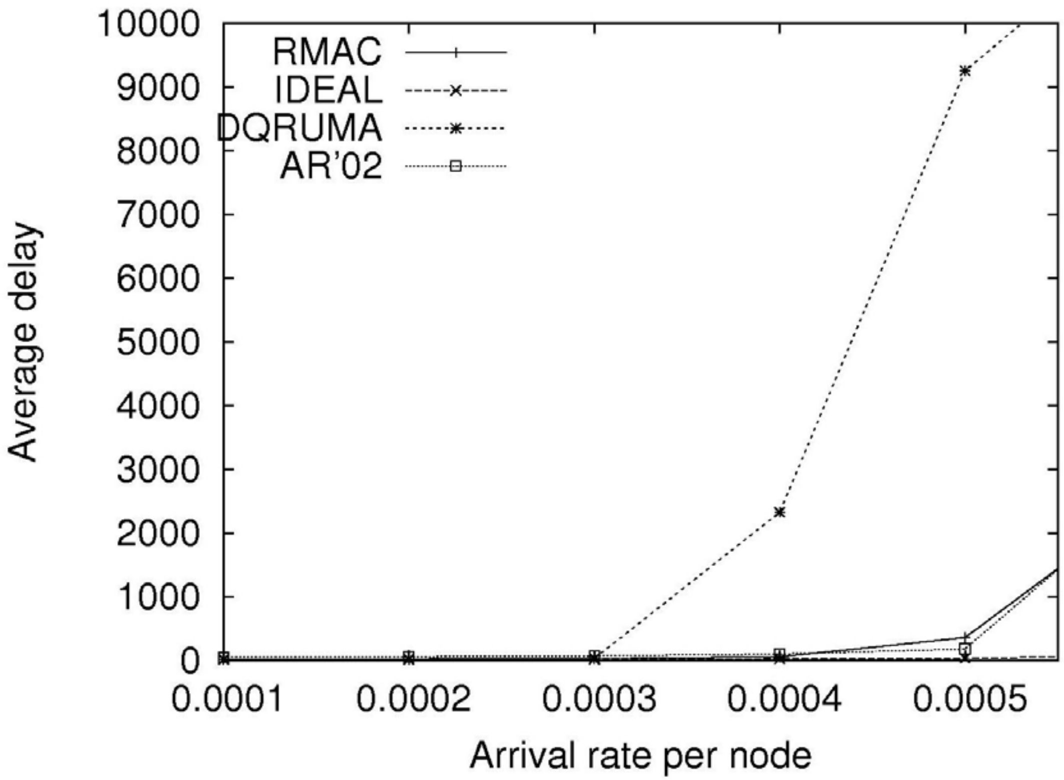
Performance - 1

- At low arrival rates



Performance - 2

- At all traffic rates



Questions?

- Are these algorithms similar? Are they making the right decisions?
- Can we do better? How?
- What about fairness?

- RMAC: a randomized adaptive medium access control algorithm for sensor networks, Suprakash Datta, SANPA, 2004.

Rationale behind protocols

- A simple model for the problem
- Use balls and bins analysis from elementary discrete math (Combinatorial analysis)
- Can we analyze the problem and the algorithms?
- First, we need (to specify) a clean mathematical model

Our model

- Network model: clustered, heterogeneous
- Node model: clock synchronization
- Traffic model:
  - Random traffic
  - Bursty traffic
  - 2 state On-OFF model
- Performance metric: delay

Analysis

- No node knows the number of sources contending for resources
- Can we estimate this number? How?
- We know
  - Number of minislots
  - Number of collisions
  - Number of idle minislots
  - Number of successful transmissions

Maximum likelihood estimation

- Given number of sources, we can find expected values of number of collisions, idle, successful minislots
- Can we solve the inverse problem?
- Our result:

$$\text{MLE}(\#\text{sources}) = n_s + 2n_c$$

What does this mean?

Maximum likelihood estimation

Intuitively:

- if MLE(#sources) > # minislots
  - Increase #minislots
  - else decrease #minislots
- RMAV: n_c (=1) > n_e (=0)
- Aboelaze et al: $n_c > n_e$
- MLE = $n_s + 2n_c > N = n_c + n_e + n_s$
- Yields $n_c > n_e$!!!

Questions?

- Are these algorithms similar? Are they making the right decisions?
- Can we do better? How?
- What about fairness?

- Need to decide on number of minislots
- Need to implement fairness

Number of minislots

- Q1: if we predict n sources how many minislots should we have?
- Q2: How do we predict the number of sources?

- Using Queueing theory, we can solve Q1. Having #minislots = #predicted packets minimizes expected delay.
- Also max throughput = 1/N(e+S+1)

Predicting number of sources

- Very difficult to do in practice
- Varies a lot with application
- We used a very simple linear model

$$p(t+1) = n(t) + a(n(t) - n(t-1)).$$

p(t) = number of minislots at time t

a = smoothing factor

n(t) = predicted #sources at time t

Dealing with fairness

- Use piggybacked requests
- The algorithm rejects piggybacked requests with probability f
- f=1: original demand assignment, most fair, bad QoS to long sessions
- f=0: bad fairness, great QoS to long sessions.
- Designer can choose the desired f

What next?

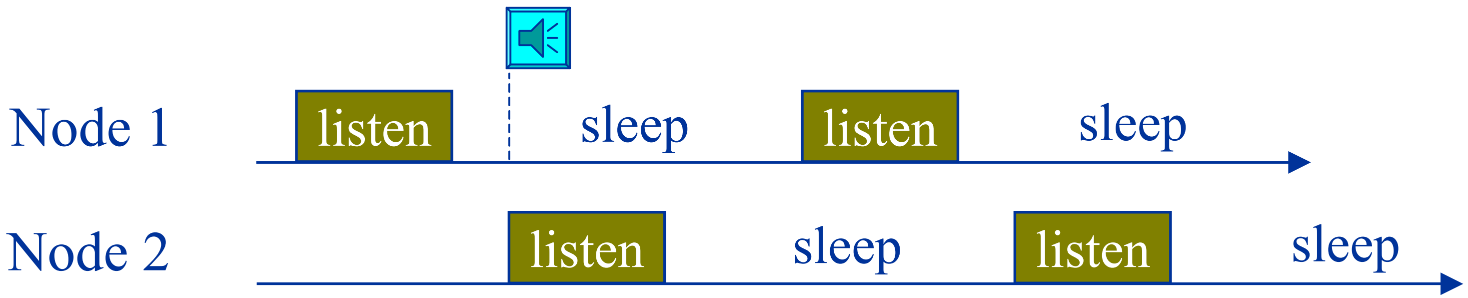
- Centralized algorithm
  - Do not need to worry about hidden terminals
- Can this be made distributed?

SMAC

- Designed for sensor networks
- slides adapted from www.cs.wmich.edu/wsn/cs691_sp03/
- Major components in S-MAC
  - Periodic listen and sleep
  - Collision avoidance
  - Overhearing avoidance
  - Message passing

Periodic Listen and Sleep

- Reduce long idle time
  – Reduce duty cycle to ~ 10% (120ms on/1.2s off)

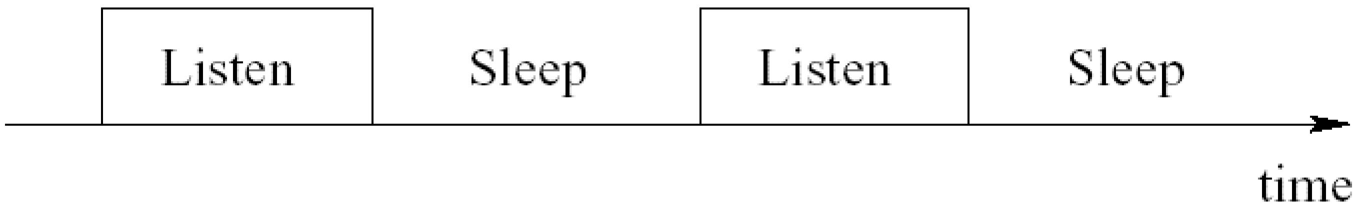


- Schedules can differ
- Prefer neighbors have same schedule
  — easy broadcast & low control overhead

Latency  Energy

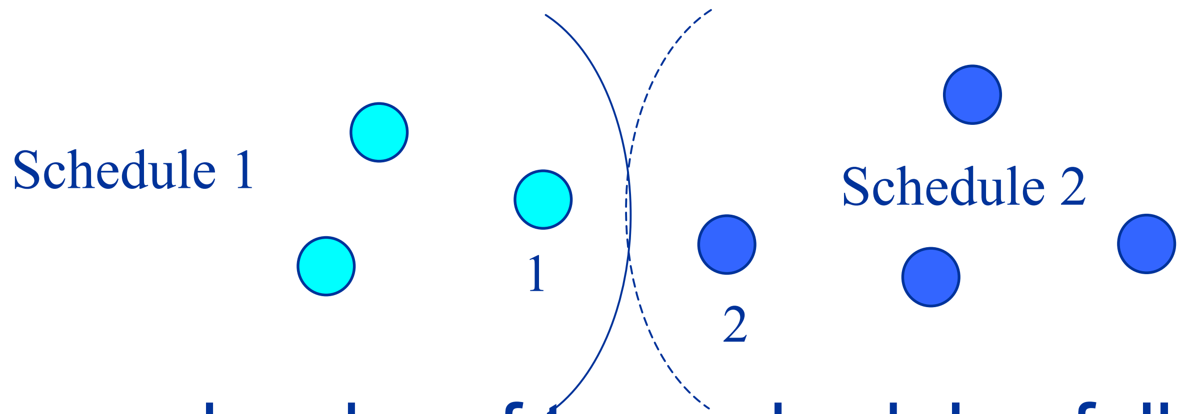
Periodic Listen & Sleep

- Nodes are in idle for a long time if no sensing event happens
- Put nodes into periodic sleep mode
  - i.e. in each second, sleep for half second and listen for other half second



Coordinated Sleeping

- Nodes coordinate on sleep schedules
  - Nodes periodically broadcast schedules
  - New node tries to follow an existing schedule



- Nodes on border of two schedules follow both
- Periodic neighbor discovery
  - Keep awake in a full sync interval over long time

Choose & Maintain Schedule

- Each node maintains a schedule table that stores schedules of all its neighbors
- Nodes exchange schedules by broadcasting them to its neighbors
  - Try to synchronize neighboring nodes together

Collision Avoidance

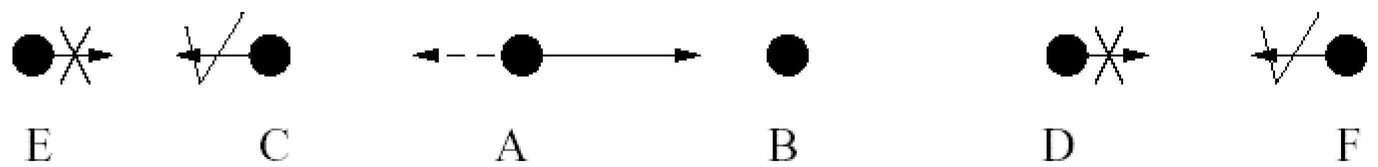
- Adopt IEEE 802.11 collision avoidance
- Virtual carrier sense
  – During field
  – Network allocation vector (NAV)
- Physical carrier sense
- RTS/CTS exchange (for hidden terminal problem)
  – Broadcast packets (SYNC) are sent without RTS/CTS
  – Unicast packets (DATA) sent with RTS/CTS

Overhearing Avoidance

- Problem: Receive packets destined to others
- Solution: Sleep when neighbors talk
  - Basic idea from PAMAS (Singh, Raghavendra 1998)
  - But we only use in-channel signaling
- Who should sleep?
  - All immediate neighbors of sender and receiver
- How long to sleep?
  - The *duration* field in each packet informs other nodes the sleep interval

Example

- Who should sleep when node A is transmitting to B?



- All immediate neighbors of both sender & receiver should go to sleep

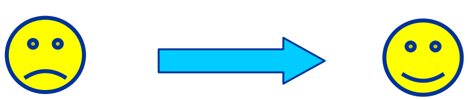
Message Passing

- How to efficiently transmit a long message?
- Single packet vs. fragmentations
  - Single packet: high cost of retransmission if only a few bits have been corrupted
  - Fragmentations: large control overhead (RTS & CTS for each fragment), longer delay
- Problem: Sensor network in-network processing requires *entire* message

Message Passing

- 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
- Other nodes sleep for whole message time

Fairness

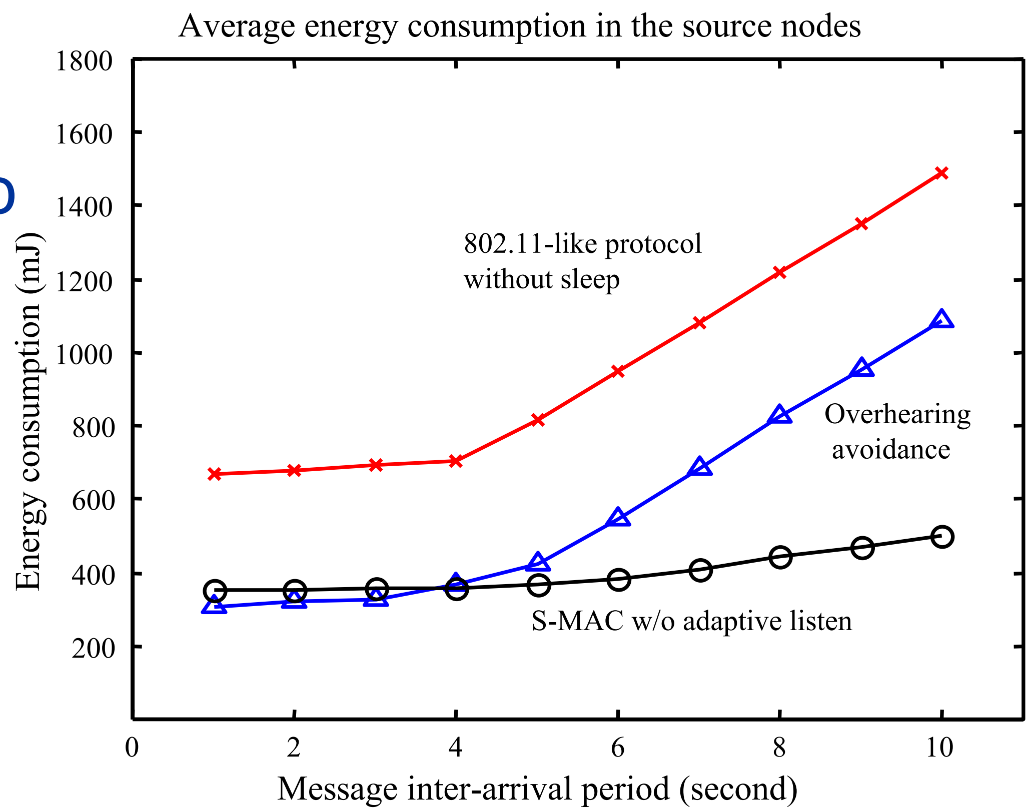


Energy
Msg-level latency

Experiment Result

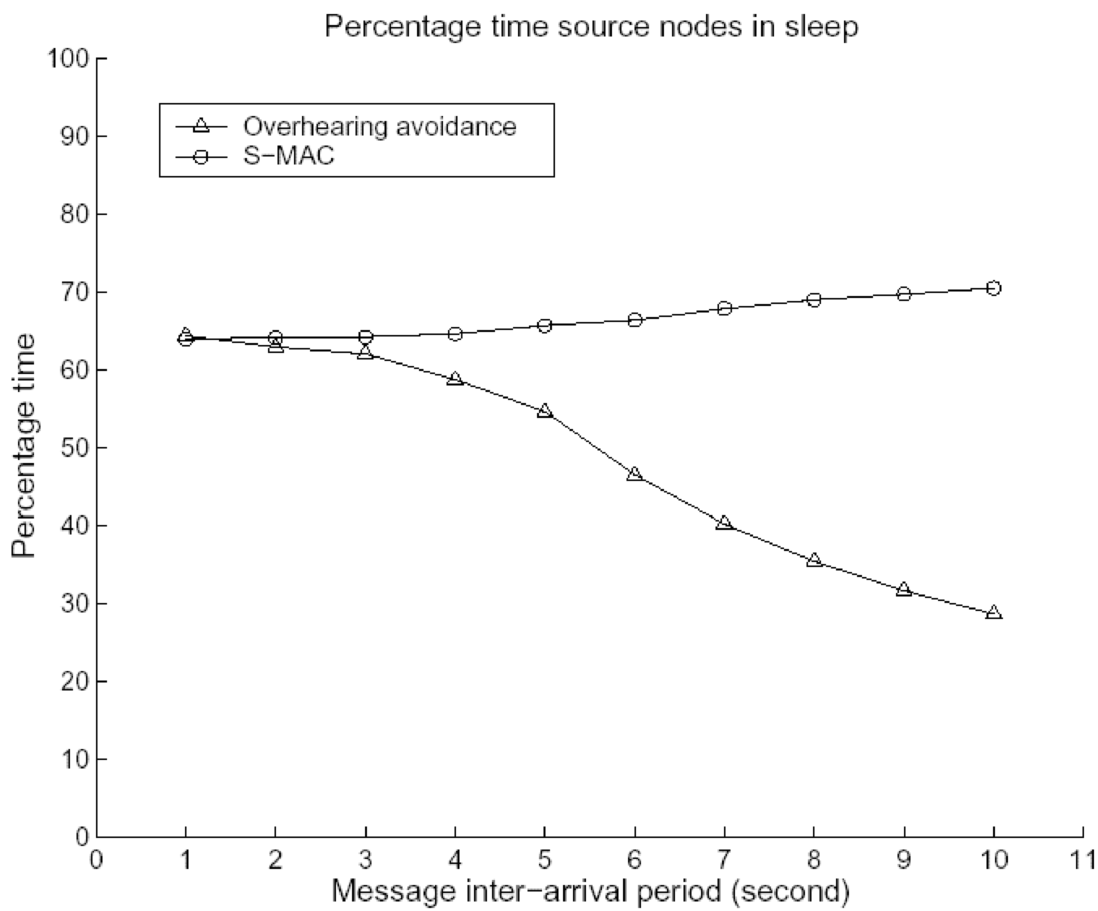
- Average source nodes energy consumption

- S-MAC consumes much less energy than 802.11-like protocol w/o sleeping
- At heavy load, overhearing avoidance is the major factor in energy savings
- At light load, periodic sleeping plays the key role



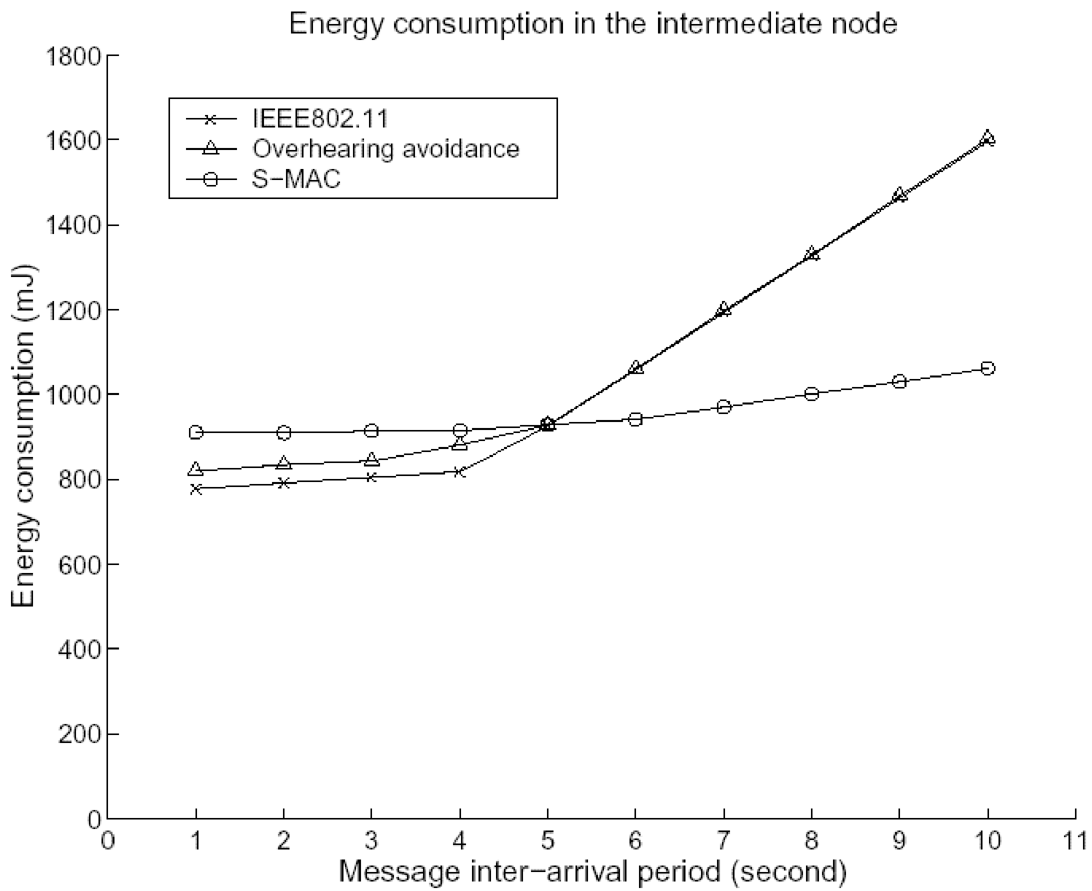
Experiment Result (contd.)

- Percentage of time source nodes in sleep



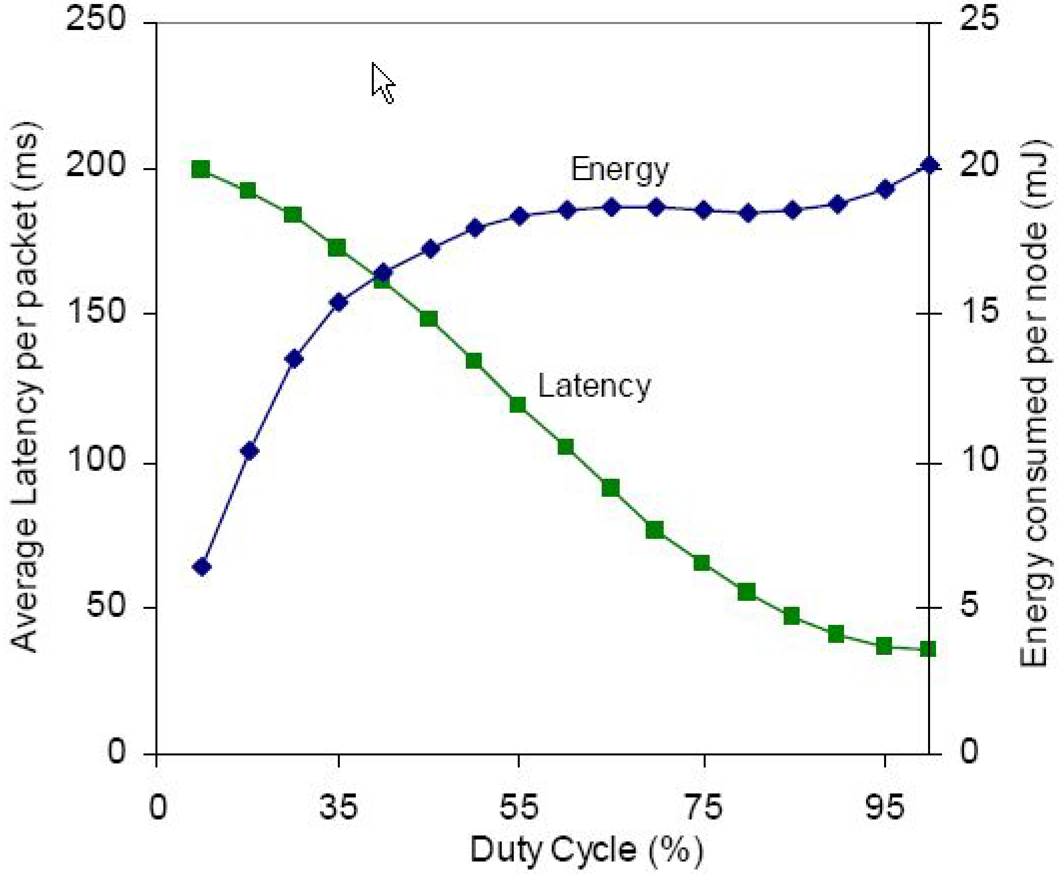
Experiment Result (contd.)

- Energy consumption in the intermediate node



Effect of duty cycle

- Important tradeoff:



S-MAC Conclusions

- Advantages:
  - Periodically sleep reduces energy consumption in idle listening
  - Sleep during transmissions of other nodes
  - Message passing reduces contention latency and control packet overhead
- Disadvantages:
  - Reduction in both per-node fairness and increase in latency