SAPIENZA
UNIVERSITÀ DI ROMA

MAC Protocols for sensing systems

Internet of Things a.a. 2019/2020

Un. of Rome "La Sapienza"

Chiara Petrioli[†]

[†] *Department of Computer Science – University of Rome "Sapienza" – Italy*

SAPIENZA
UNIVERSITÀ DI ROMA

Energy-efficient MAC protocols

Internet of Things a.a. 2019/2020

Un. of Rome "La Sapienza"

Chiara Petrioli[†]

[†] *Department of Computer Science – University of Rome "Sapienza" – Italy*

SAPIENZA
UNIVERSITÀ DI ROMA

S-MAC (Sensor MAC)

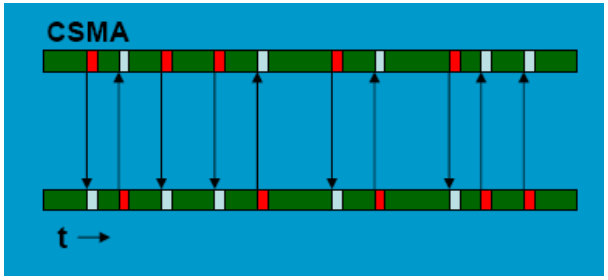
W. Ye, J. Heidemann, D. Estrin “An energy efficient MAC Protocol for Wireless Sensor Networks”, IEEE Infocom 2002

Synchronized MAC based on duty cycle

Performance objectives

1) Energy efficiency

- Sources of energy waste
  - *collision*. When a transmitted packet is corrupted it has to be discarded, and the follow-on retransmissions increase energy consumption. Collisions also increase latency.
  - *overhearing*, meaning that a node picks up packets that are addressed to other nodes.
  - *control packet overhead*
  - *idle listening*, *i.e.*, listening to receive possible traffic that is not sent (major source of energy consumption).



2) End-to-end latency

3) Fairness

4) Network capacity/scalability (to density and traffic)

Nodes sleeping scheme

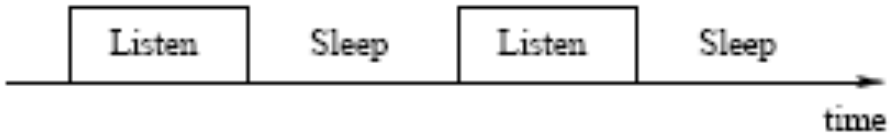


Fig. 1. Periodic listen and sleep.

- Nodes follow an awake/asleep schedule with a given duty cycle d
- In S-MAC nodes schedule are synchronized
  - all nodes transmit in the same slot and receive in the same slot if possible **(WHICH ARE THE PROS? WHICH ARE THE CONS?)**

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

  - Periodic exhange of information is needed to resynch in case of clock drifts (if resynch every few tens seconds drifts remain limited)

Choosing and maintaining schedules

- Before a node starts its periodic «listen and sleep» activity, it needs to choose a schedule and broadcast it to its immediate neighbors (schedule exchange).
  - at start up node x listens for some random time
    - ✓ if x receives a SYN from another node y, it synchronizes to its schedule (x is a *follower*). It waits for a random delay t_d and rebroadcasts its schedule.
      - follower of the same synchronizer do not collide (or collide with low prob) thanks to t_d
    - ✓ otherwise node x selects a random time T to sleep before waking up again and sends this value T to neighbors in a SYN (x therefore becomes a *synchronizer*)
    - ✓ if a node receives a different schedule after it selects its own, which is followed by more than one device, it adopts both schedules, broadcasting the new one
      - "border nodes" where two synch waves meet are the ones with multiple schedules
        - » they consume more energy
- Each node also maintains a *schedule table* that stores the schedules of all its known neighbors.

When a node has a packet to send …

- It waits for the destination to be ON and sends the packet following CSMA/CA
  - performs carrier sense for a random interval
  - if no transmission within this interval the floor is taken (physical carrier sense) to transmit RTS/CTS
  - if the RTS/CTS is successful (virtual carrier sensing) DATA is sent which is followed by an ACK
  - NAVs are used for deciding for how long nodes should go to sleep before they can try to access again in case neighbors are transmitting
  - to better exploit the time needed to hanshake (RTS/CTS) bursts of packets are transmitted if more packets are in queue for the same destination
    - ✓ Limited packet size and transmission of ACKs following reception avoids hidden terminal problem if nodes waking up wait for some limited time before transmitting

SAPIENZA
UNIVERSITÀ DI ROMA

Maintaining synchronization

- Some initially exchanged SYN maybe lost e.g. due to collision, or new nodes maybe added
- Clock drifts

How do we keep nodes schedules up to date and synchronized?

- A node periodically sends a SYN.
- For nodes to receive SYN and DATA listen times are divided into two intervals

Maintaining synchronization

- Some initially exchanged SYN maybe lost e.g. due to collision, c
- Clock drif

How do we k and synchroni

- A node p
- For node times are divided in

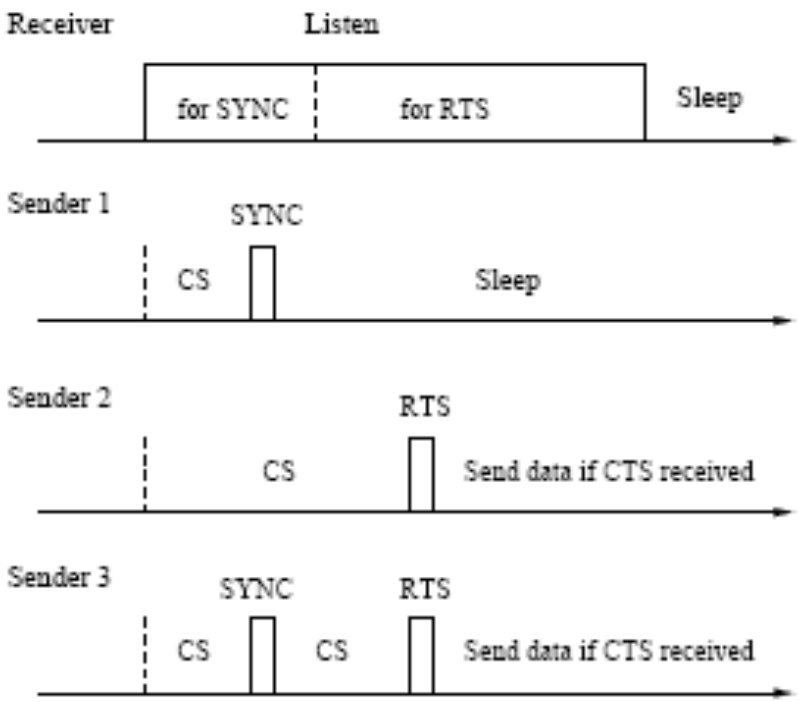


Fig. 3. Timing relationship between a receiver and different senders. CS stands for carrier sense.

S-MAC: Limits

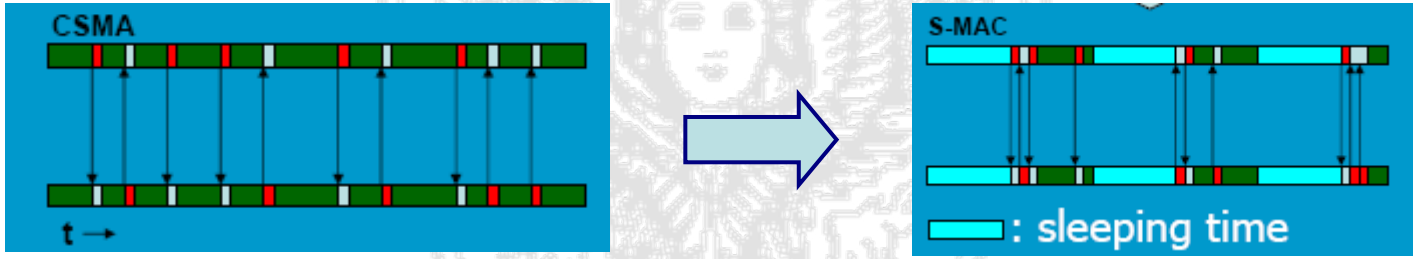
- Needs synchronization
  - even if clock drifts are not a major problem synchronization adds control overhead which may impair long lifetimes (e.g., in those applications where communication needs are sporadic)
- Throughput is reduced since only the active part of the frame is used for communication
  - It is further reduced because tx/rx occur only during scheduled ON times (that are all synchronized)
- Latency increases since when a node generates a packet it has to wait for the next hop relay on time before the packet can be forwarded.

T-MAC (Timeout MAC)

Tijs van Dam, Koen Langendoen “An adaptive energy efficient MAC Protocol for Wireless Sensor Networks”, ACM SenSys 2003

Synchronized MAC based on duty cycle

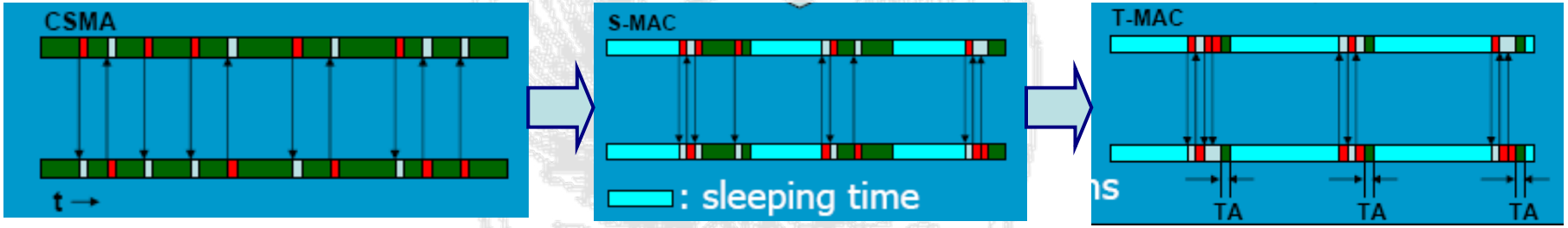
- Observation: In SMAC there are two critical parameters (the active time and the frame time)
  - a long frame time increases latency
  - given an active time the longer the frame time the lower the energy consumption
  - the active time should be dimensioned based on traffic: for a frame time the higher the traffic, the longer the active time should be



  - In SMAC the two parameters are fixed
    - ✓ setting should depend on worst case
  - in TMAC the frame time is fixed but the active time is dynamically adapted

T-MAC

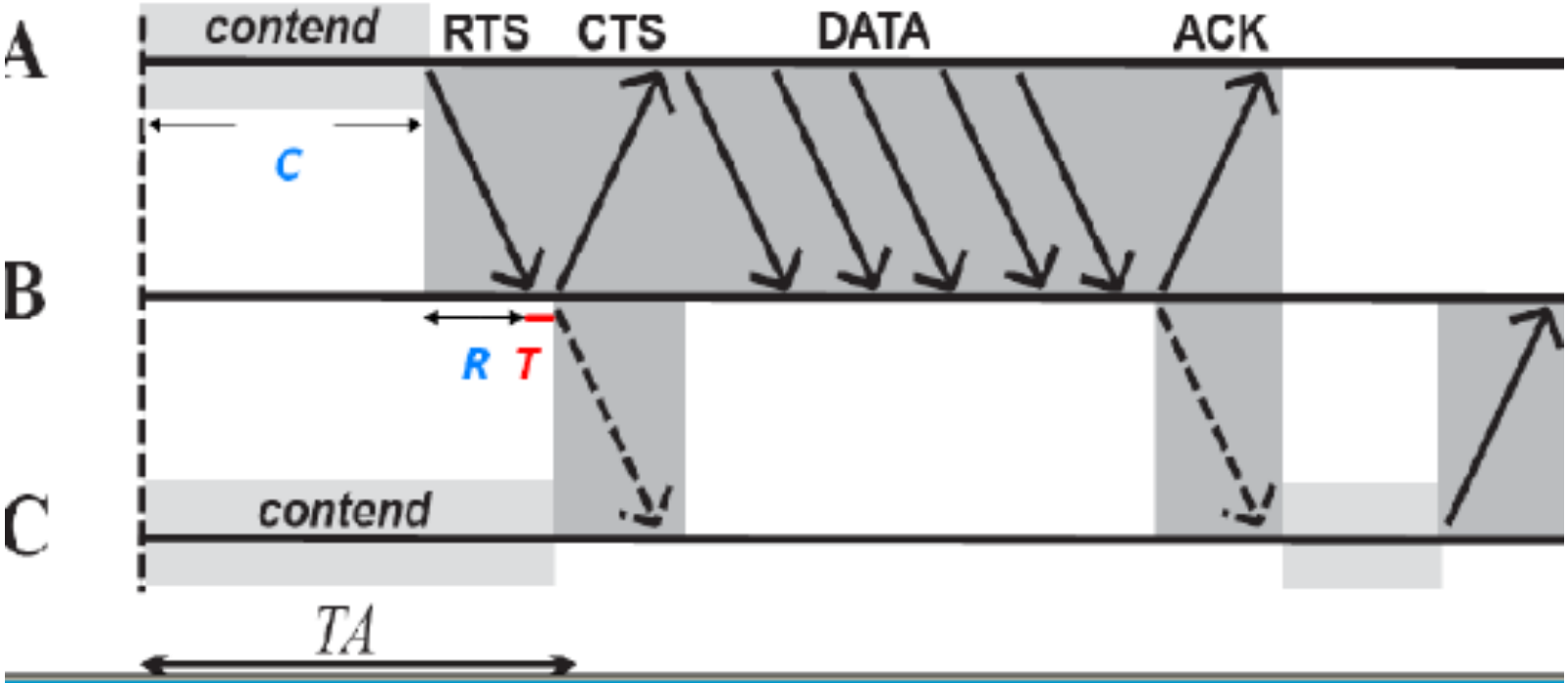
- Nodes synchronize their schedules using the SMAC virtual clustering approach.
- Within an active time CSMA/CA + back to back packet transmission in bursts are adopted
- Changes from S-MAC: if no transmission from neighbors for a time TA the active time is aborted and node goes to sleep



- TA timer is reset if: a) any data is received on the radio,b) communication (e.g, collision) is sensed on the radio, c) data are transmitted, d) RTS/CTS are exchanged by neighbors
  - ✓ A node should not go to sleep while its neighbors are still communicating since it maybe the receiver of a subsequent message

How to determine TA

Determining of TA



- TA > C+R+T (must be long enough to receive at least the start of the CTS packet)

- other changes from SMAC:
  - When a node sends an RTS but does not receive a CTS back this may be due to one of the following events:
    - ✓ 1) the RTS was not received due to collisions
    - ✓ 2) the receiving node cannot answer due to an RTS/CTS overheard
    - ✓ 3) the receiving node is sleeping

    In cases 1-2) reducing the active time would be wrong
      - " a node should retry by resending the RTS at least twice before giving up and going to sleep"
  - early sleep may degrade throughput (while decreasing idle listening and energy consumption)
    - ✓ mechanisms introduced to signal to nodes there is traffic for them at the beginning of the active time to prevent them from going to sleep

Early sleep problem

Data gathering at the sink

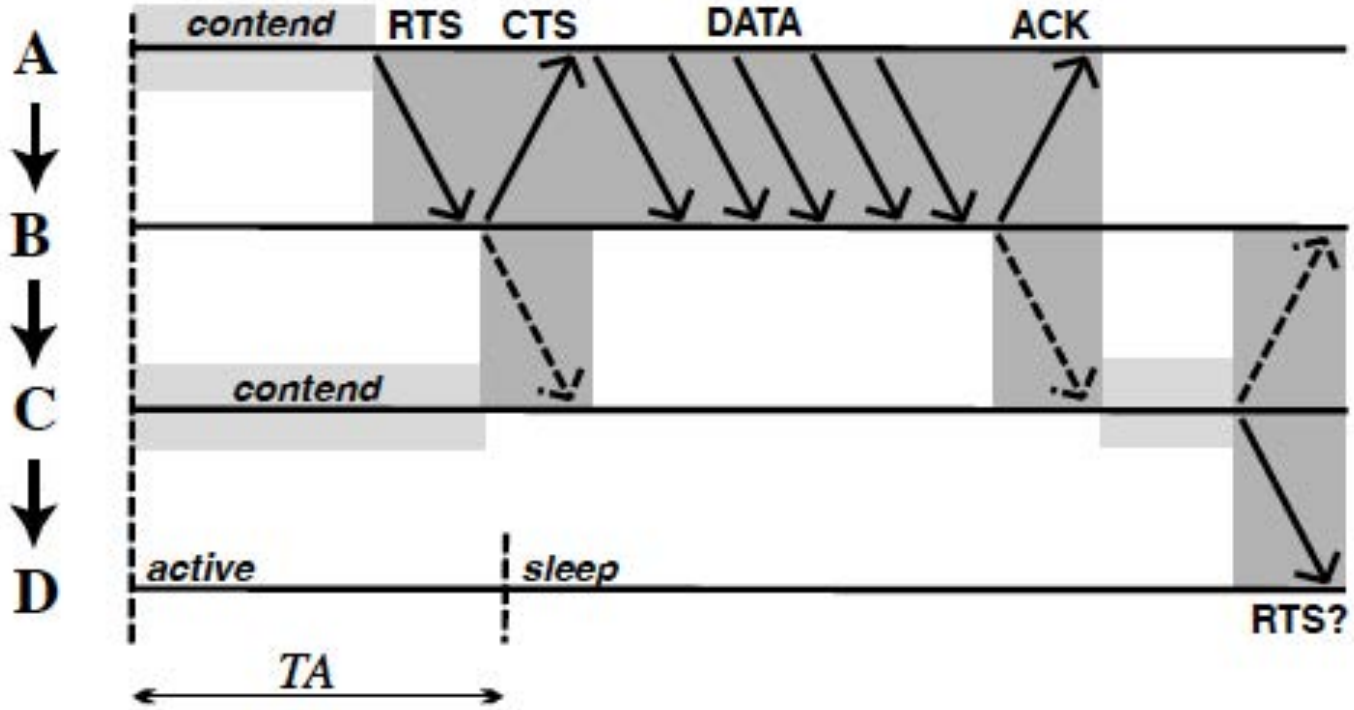


Figure 4: The *early sleeping* problem. Node D goes to sleep before C can send an RTS to it.

Early sleep- Solution 1

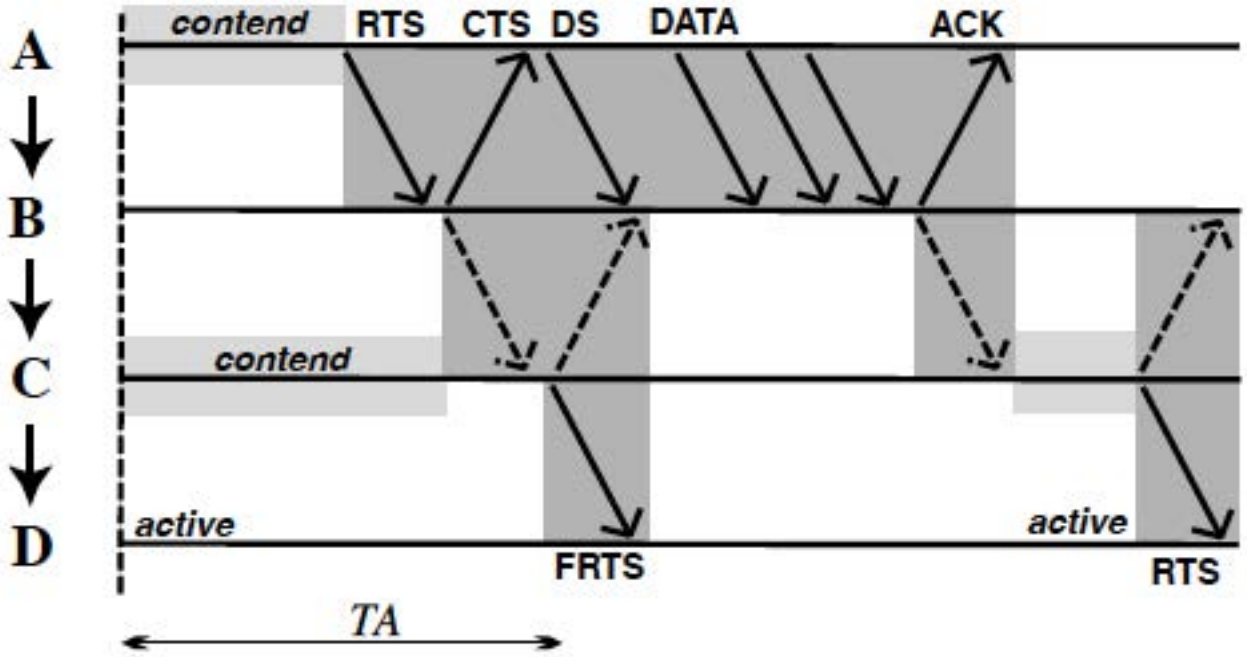


Figure 5: The future-request-to-send packet exchange keeps Node D awake.

The sender must wait before transmitting the real data that a FRTS is received. To maintain the channel floor in the meanwhile it transmits a dummy DS (Data Send) packet

Early sleep- Solution 2

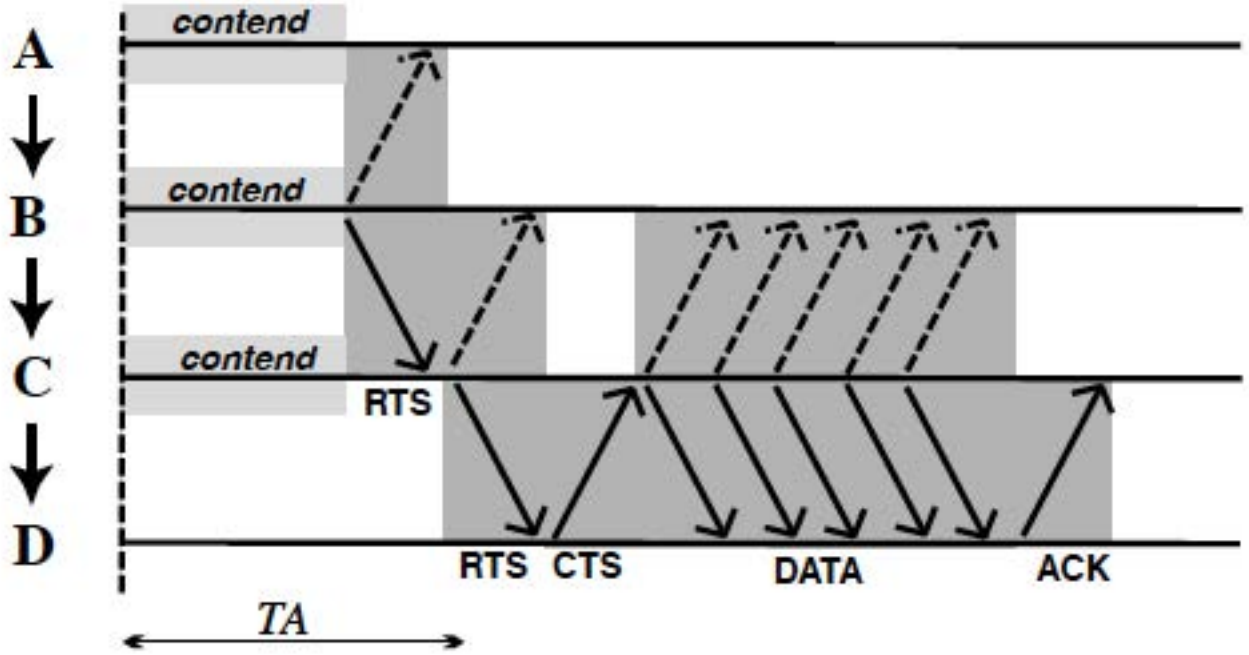
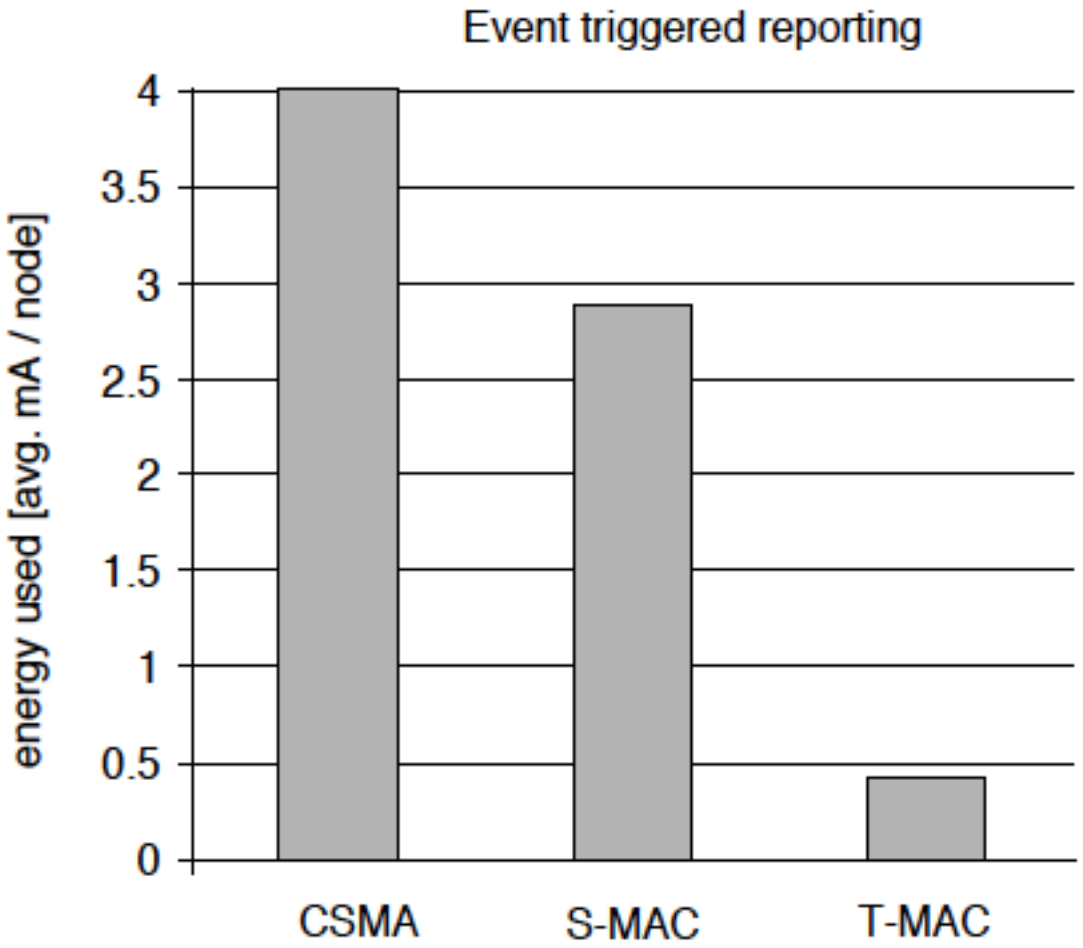


Figure 6: Taking priority upon receiving RTS.

Full buffer priority: upon receiving an RTS a node which has almost the buffer full instead of answering with a CTS sends immediately an RTS

T-MAC performance



SAPIENZA
UNIVERSITÀ DI ROMA

BMAC (Berkeley Media Access Control)

- Polastre, Hill, Culler “Versatile Low Power Media Access for Wireless Sensor Networks” , ACM SenSys 2004

- Asynchronous MAC

Targets

- The MAC
- Should have low Power Operation
- Should perform effective Collision Avoidance
- Simple Implementation, Small Code and RAM Size
- Efficient Channel Utilization at Low and High Data Rates
- Reconfigurable by Network Protocols
- Tolerant to Changing RF/Networking Conditions
  - links can be dynamic
- Scalable to Large Numbers of Nodes

B-MAC

- For effective collision avoidance, a MAC protocol must be able to accurately determine if the channel is clear—Clear Channel Assessment or CCA
  - BMAC proposes a way to estimate the channel noise and to determine whether the channel is free (taking some samples and checking whether any of the sample is below the average noise level)
    - ✓ the proposed solution for channel assessment has been validated with experimental data
    - ✓ queue of RSSI samples (10), median of the samples used to compute an exponentially weighted moving average with decay factor alpha (0.06) →noise floor estimation
    - ✓ CCA samples→if no outlier out of 5 samples (outlier= below noise level) then busy; otherwise free

B-MAC

RECEIVER SIDE

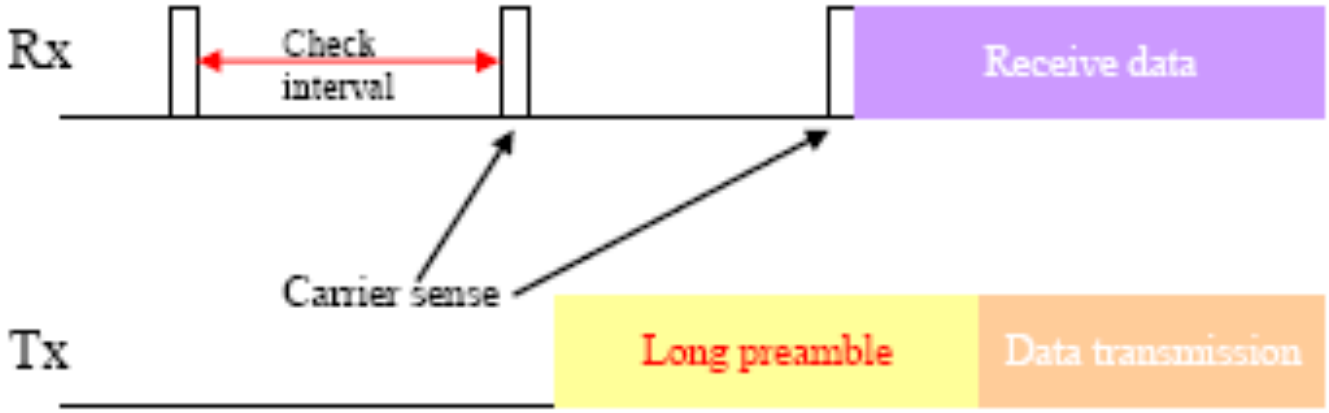
- B-MAC duty cycles the radio through periodic channel sampling, called Low Power Listening (LPL)
  - Each time the node wakes up, it turns on the radio and checks for **activity**. If activity is detected, the node powers up and stays awake for the time required to receive the incoming packet. After reception (or after a timeout expiration), the node returns to sleep.

TRANSMITTER SIDE

- The sender transmits a preamble, then the data
  - To reliably receive data, the preamble length is matched to the interval that the channel is checked for activity

B-MAC

Shifts most burden to the sender



Challenge

Check interval has to be short to ensure reasonable size preambles

XMAC

Buettner,Yee,Anderson, Han “X-MAC: A short preamble MAC protocol for duty cycled wireless sensor networks”, ACM SenSys 2006

Starting point for XMAC

- A key advantage of asynchronous low power listening protocols such as BMAC is that sender and receiver can be completely decoupled in their duty cycles
  - no need for synchronization
- BMAC long preamble in low power listening however leads to performance degradation
  - the receiver has to wait for the full period until the preamble is finished before the data/ack exchange can begin, even if the receiver has woken up at the start of the preamble
    - ✓ increase in latency and energy consumption
  - overhearing problem
    - ✓ **receivers who are not in the target of the sender also wake up during the long premable and have to stay on until the end of it to discover they are not the intended destination**
      - Increase in energy consumption!
  - latency degradation
    - ✓ per hop latency lower bounded by preamble length

XMAC

- Ideas
  - embed address info of the intended destination in the preamble
    - ✓ to avoid overhearing
  - use a *strobed preamble* : the preamble is a series of short preambles. Pauses between the short preambles allow the destination to send a fast ACK when up
    - ✓ reception of an early ACK makes the sender stop sending short preambles
      - the preamble is automatically set to the right size

This approach also solves a practical problem ← packet radios not always able to send long preambles

XMAC

- Idea
  - en … preamble
    - ✓
  - us … ort
    pre … he
    des …
    - ✓ … short

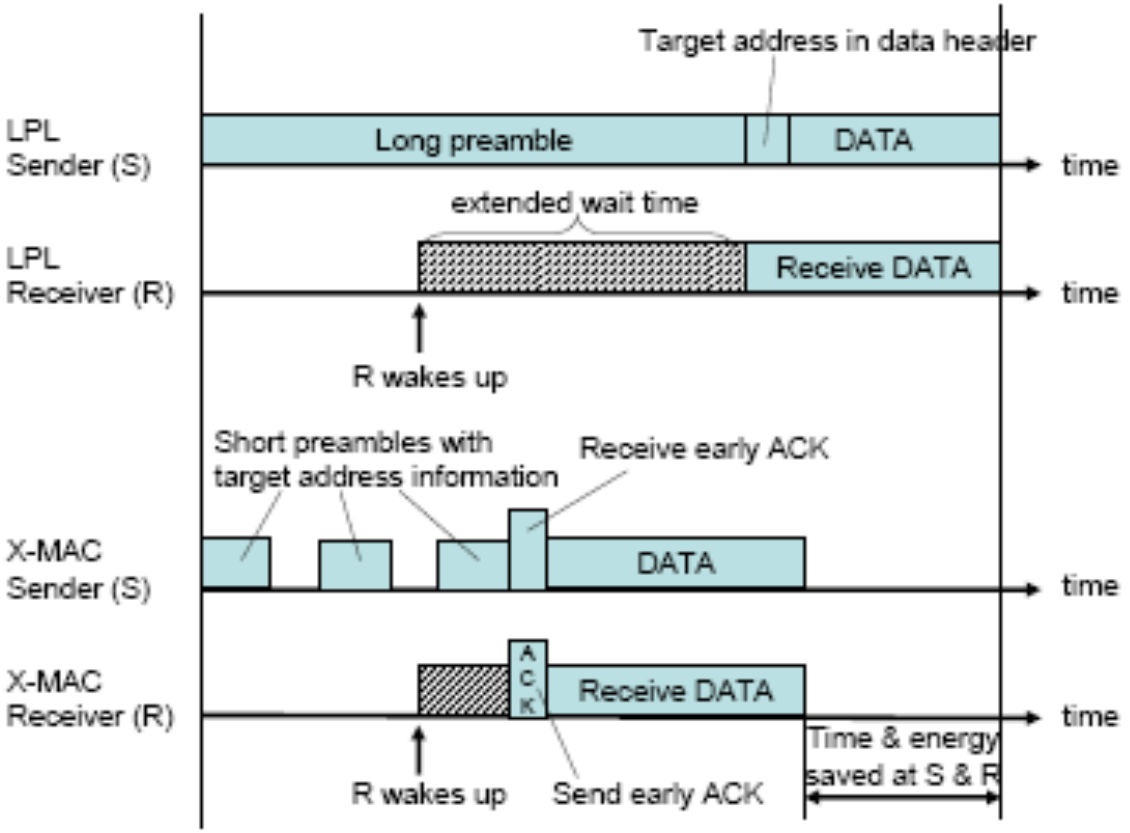


Figure 1. Comparison of the timelines between LPL's extended preamble and X-MAC's short preamble approach.

X-MAC Perf. Evaluation

- star topology, 9 sending nodes, each transmitting on average one packet per second, 500ms preamble

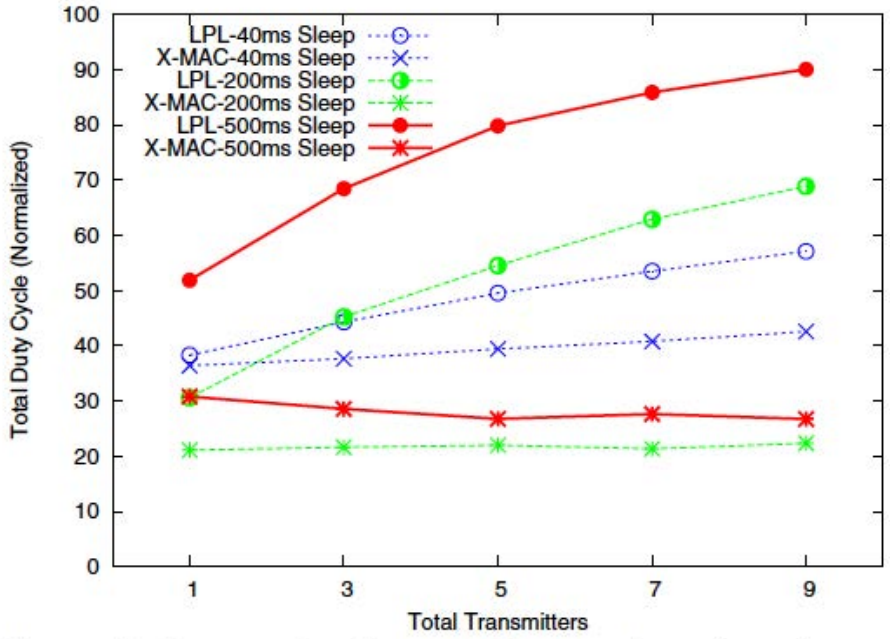


Figure 9. Duty cycle of contending senders, 1 packet per second.

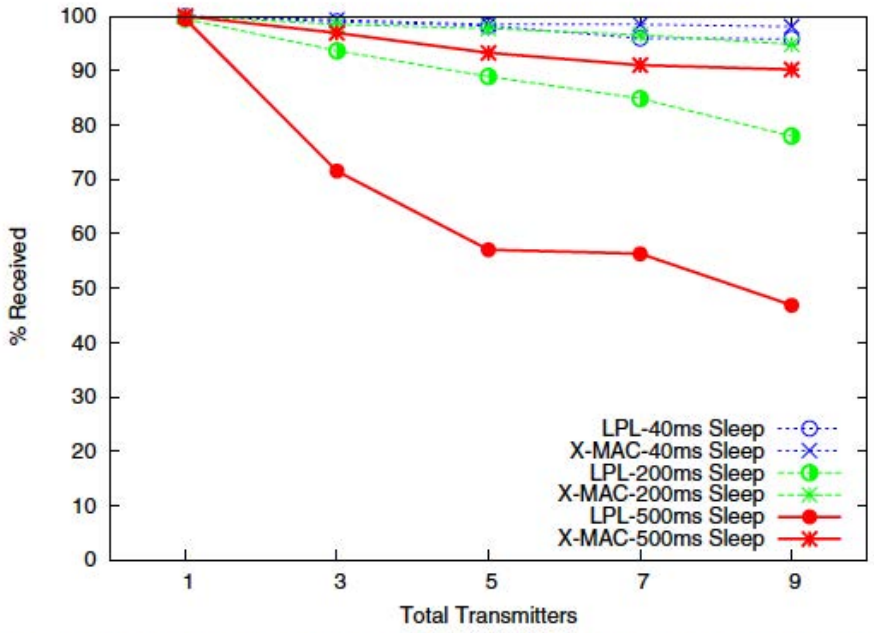


Figure 12. Reception success rate, 1 packet per second.

WiseMAC

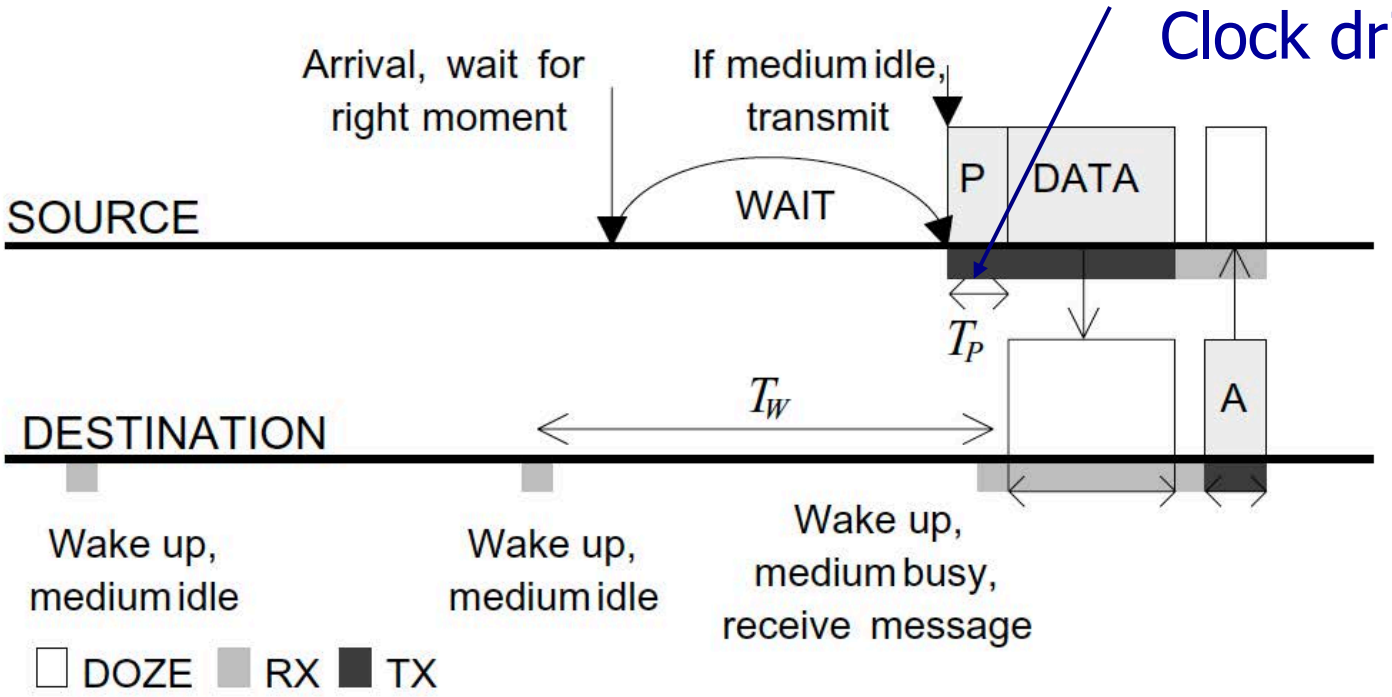
- Amre El-Hoiydi and Jean-Dominique Decotignie "WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks", in proceedings of IEEE ISCC 2004

- Asynchronous MAC

WiseMAC

Accounts for

Clock drifts



Let θ be the frequency tolerance of the time-base quartz

SAPIENZA
UNIVERSITÀ DI ROMA

Clock Drift

- Crystal Clock accuracy is defined in terms of ppm (ppm= parts per million).
- What is the error you will encounter when using a clock of a specific type with a specific accuracy?
- Realistic example: RTC clock on an IoT device with a drift around 10ppm.
- 10ppm=10/10e6=10-5
- 86400 seconds in a day
- Total error of 86400*10-5=0,8s per day
- This translates in a max drift of about 300 seconds for 6 months of operation (each clock +/- θ according to the ppm error).