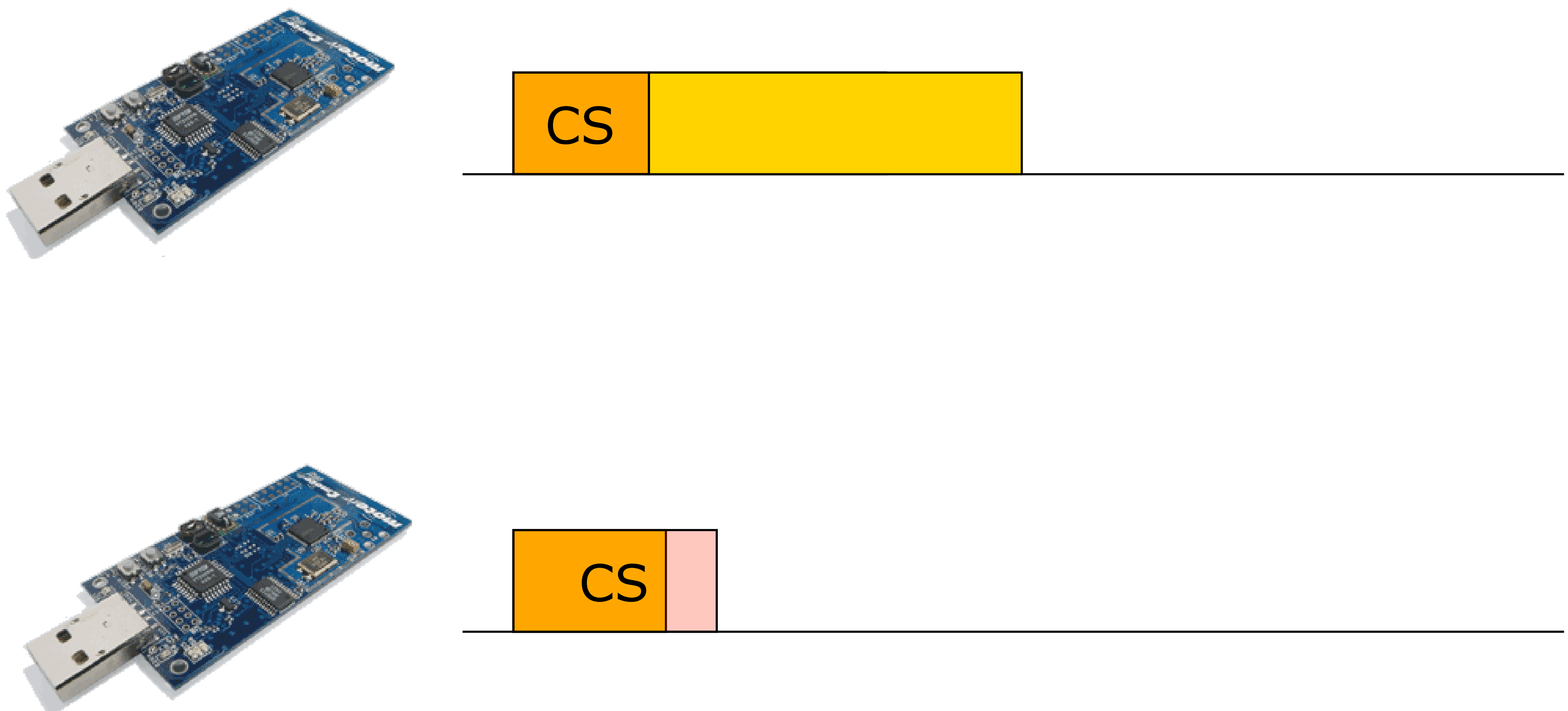
Media Access Control

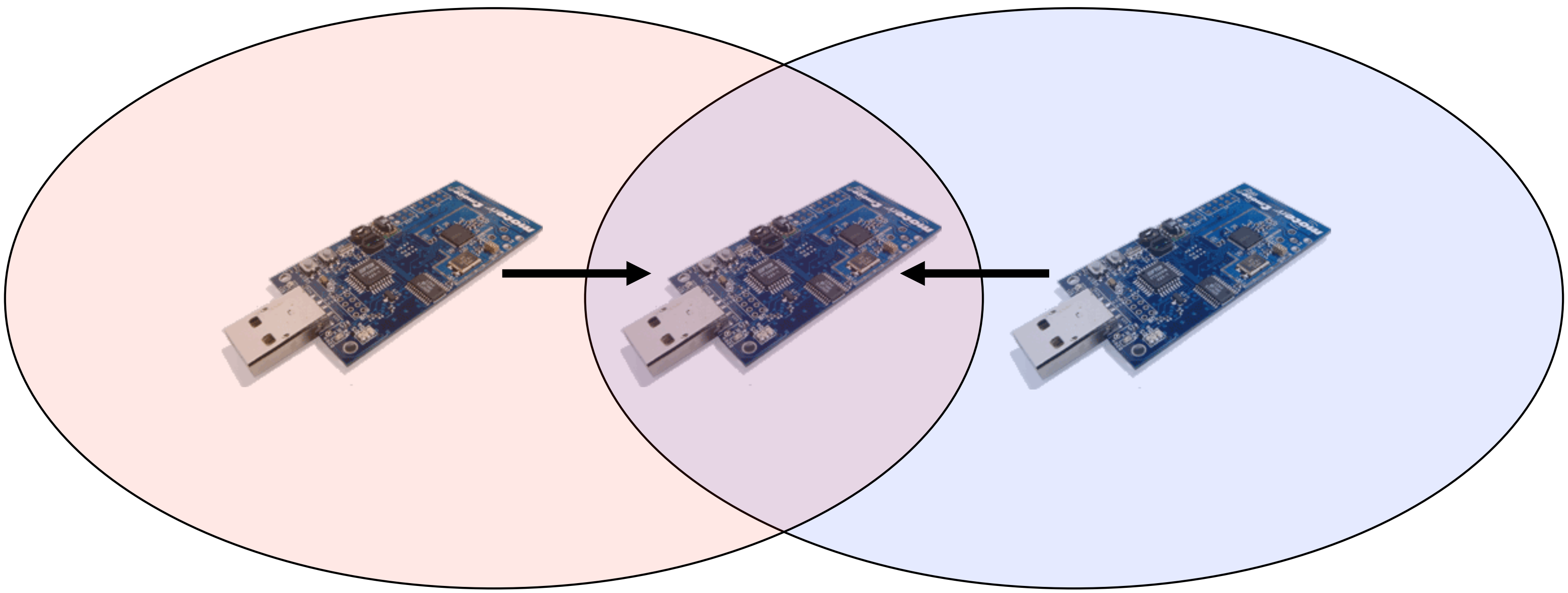
Slides by Chenyang Lu

CSMA/CA

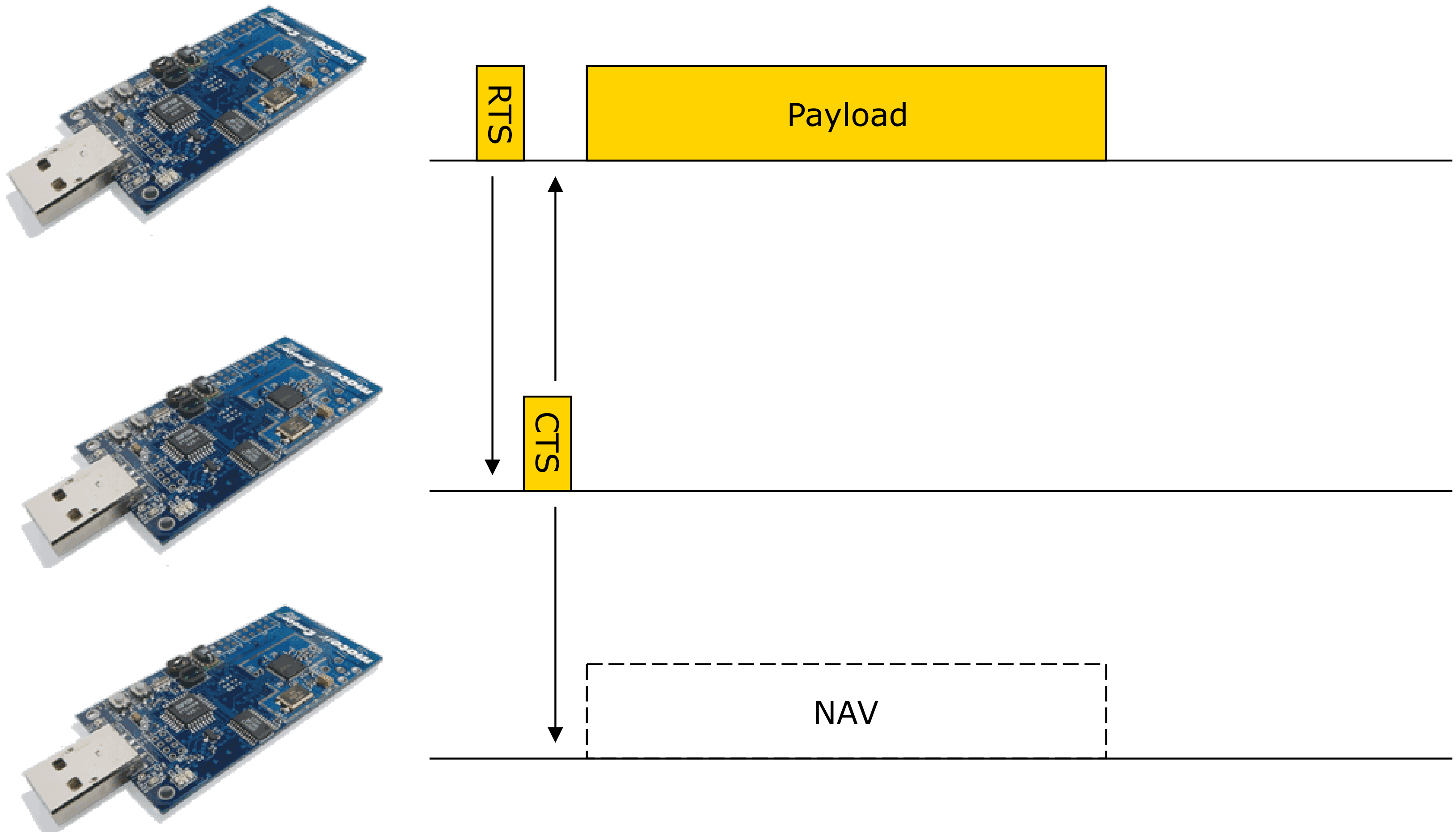
Carrier Sense Multiple Access, Collision Avoidance



Hidden Terminal Problem



Virtual Carrier Sense: RTS/CTS



Power-Saving MAC

- Many applications' expected lifetime: months or years
- Actual lifetime:
  - AA batteries: 2000 mAh (if you're lucky)
  - CC2420 radio: 19.7 mA when idle but awake (RX mode)
  - 2000 mAh / 19.7 mA = 101.5 h ≈ 6 days

- This is a problem!
- Solution: keep the radio asleep most of the time
  - Duty cycles on the order of 0.1% – 1%

Types of Power-Saving MACs

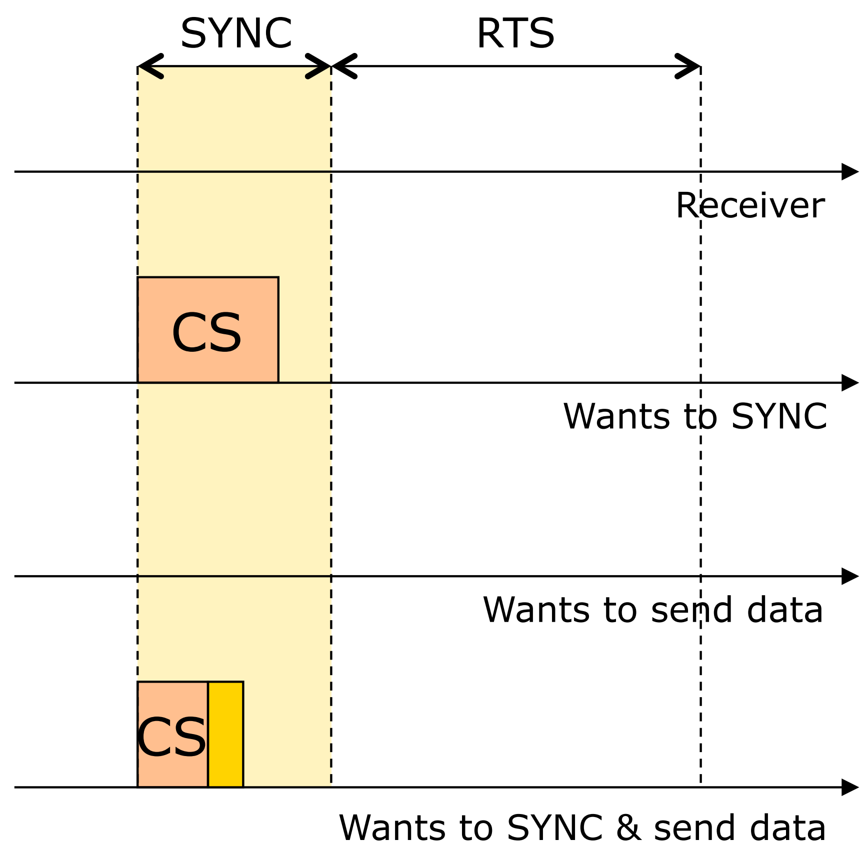
- Scheduled contention: nodes periodically wake up in unison, contend for access to channel, then go back to sleep
  - **S-MAC** [Ye 2002], T-MAC [van Dam 2003]
- Channel polling: nodes independently wake up to sample radio channel
  - **B-MAC** [Polastre 2004], **X-MAC** [Buettner 2006]
- Time Division Multiple Access (TDMA): nodes maintain schedule of when to wake and when they're allowed to transmit
  - DRAND [Rhee 2006]
- Hybrid protocols: **SCP** [Ye 2006], **Z-MAC** [Rhee 2005], Funneling MAC [Ahn 2006], 802.15.4 [IEEE 2003],

S-MAC: Synchronized Sleeping

- Nodes stay asleep most of the time
- Periodically wake up for short intervals to see if anyone's sending
- Low energy consumption when traffic is low
- Sacrifice latency for longer lifetime

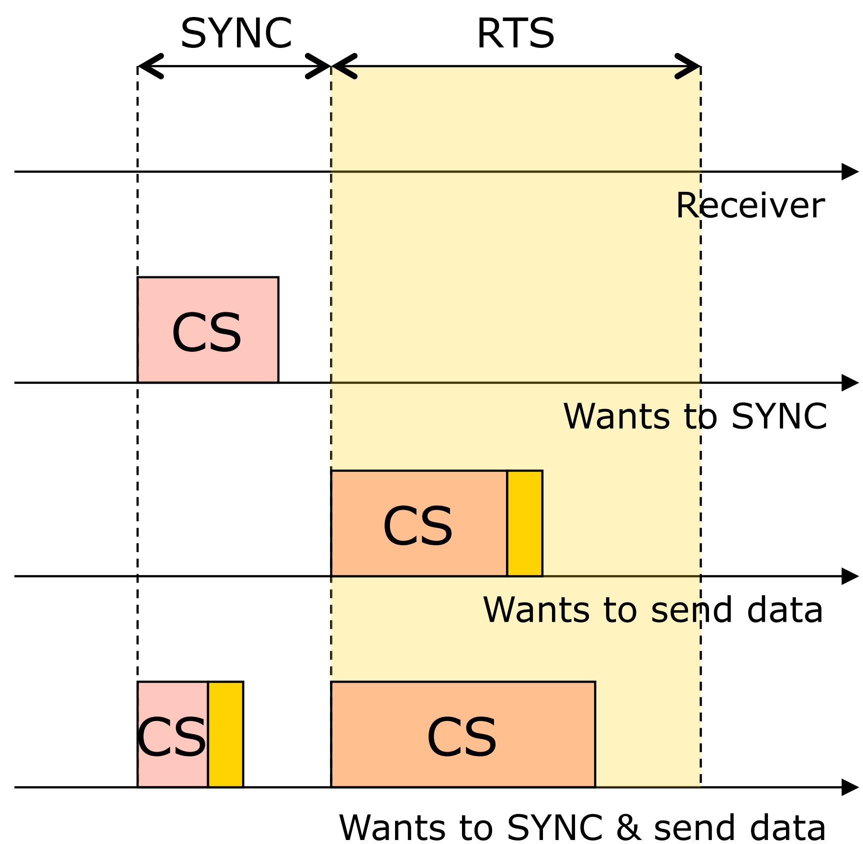
S-MAC: Sending a Packet

- Time awake divided into two parts: SYNC and RTS
- Node periodically send SYNC packet to keep clocks in sync
- CSMA/CA used to contend for access to wireless channel



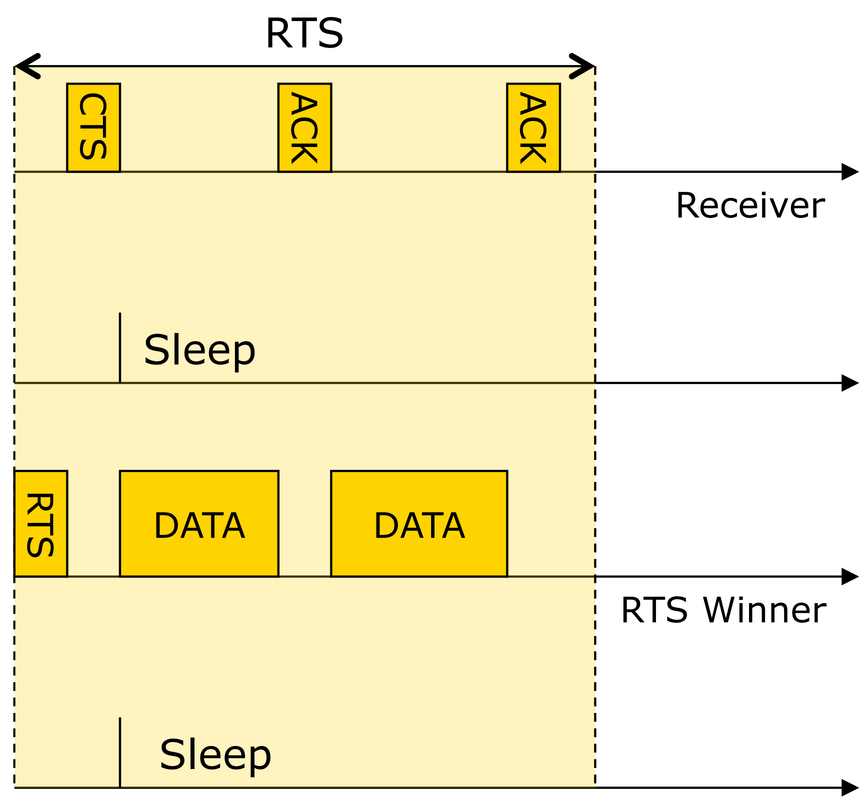
S-MAC: Sending a Packet

- RTS section used for transmitting data
- CSMA/CA again, followed by RTS/CTS



S-MAC: Sending a Packet

- CTS for someone else → go to sleep
  - Overhearing avoidance
- Sender does one RTS/CTS then sends data for rest of frame
  - Application performance >> node-level fairness
- All data packets are ACKed
  - Packet fragmentation = higher reliability

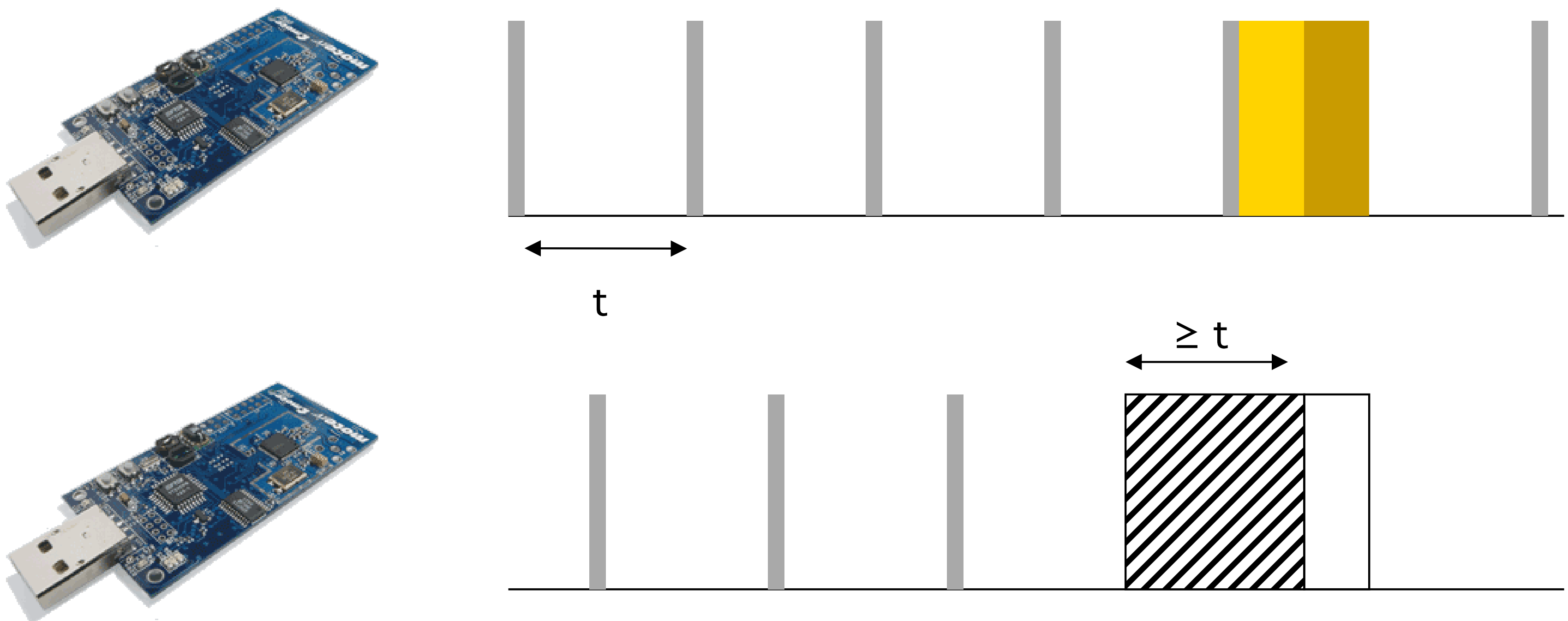


A Look at S-MAC

- Power savings over standard CSMA/CA MAC
- Long listening interval is expensive
  - Everyone stays awake unless somebody transmits

- Time sync overhead even when network is idle
- RTS/CTS and ACK overhead when sending data

B-MAC

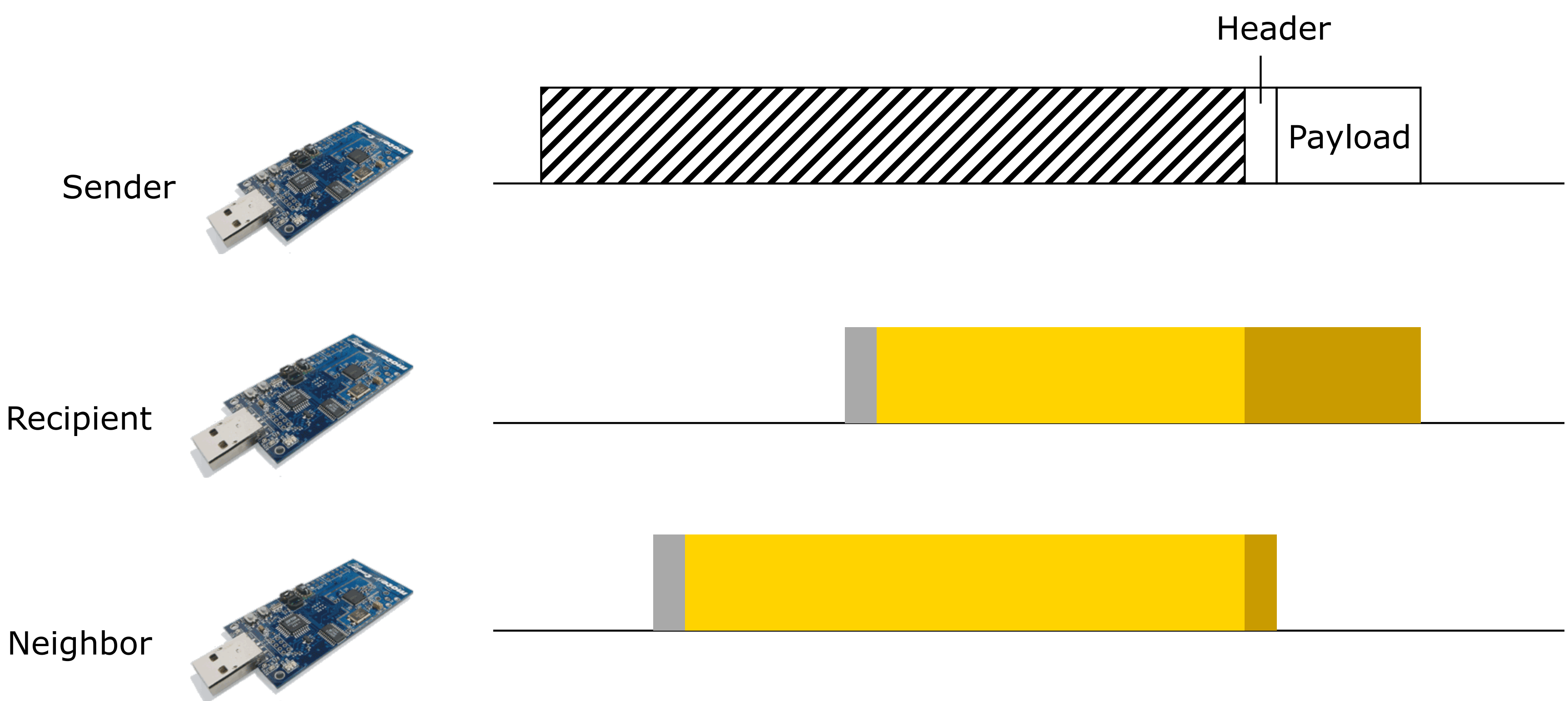


A Look at B-MAC

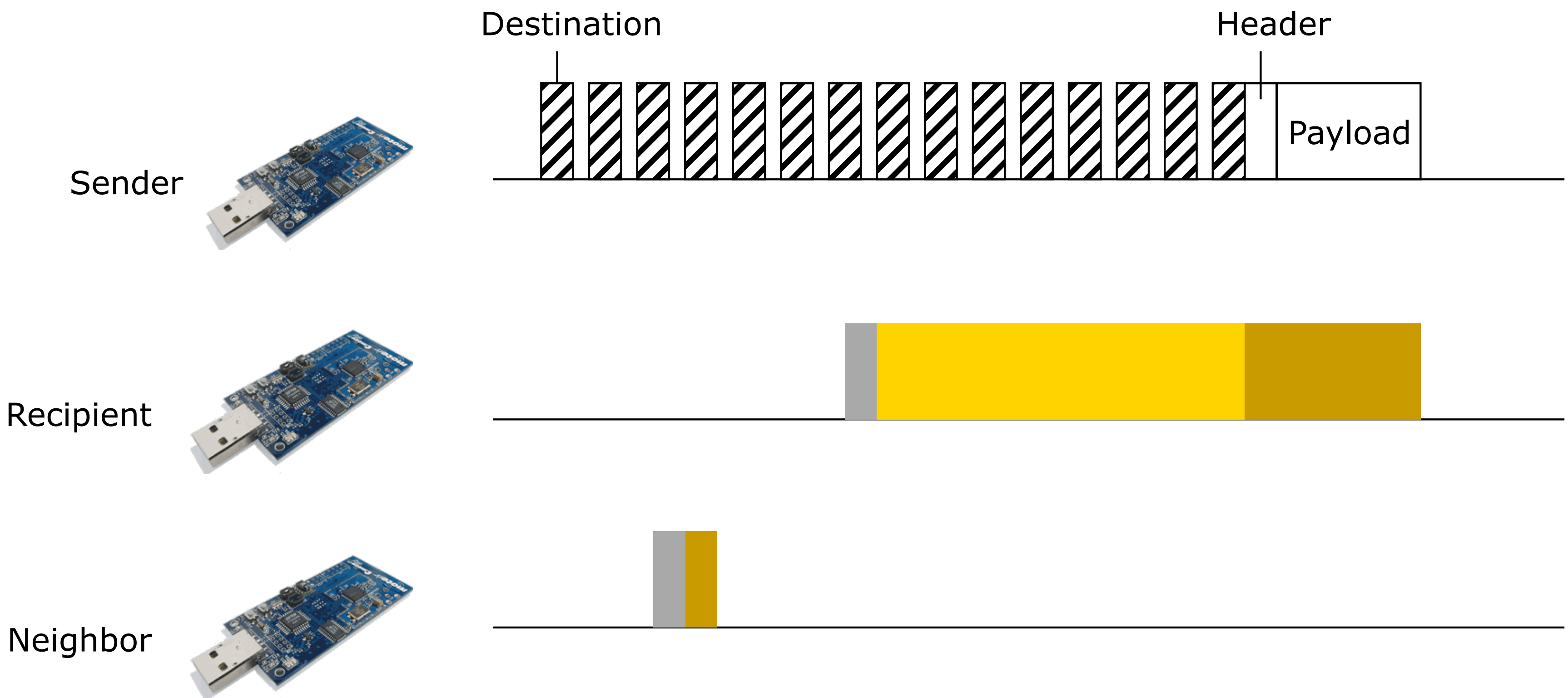
- Low overhead when network is idle
- Simple to implement
- Better power savings, latency, and throughput than S-MAC

- Lower duty cycle → longer preambles:
  - Higher average latency
  - Higher cost to send
  - Higher cost to overhear

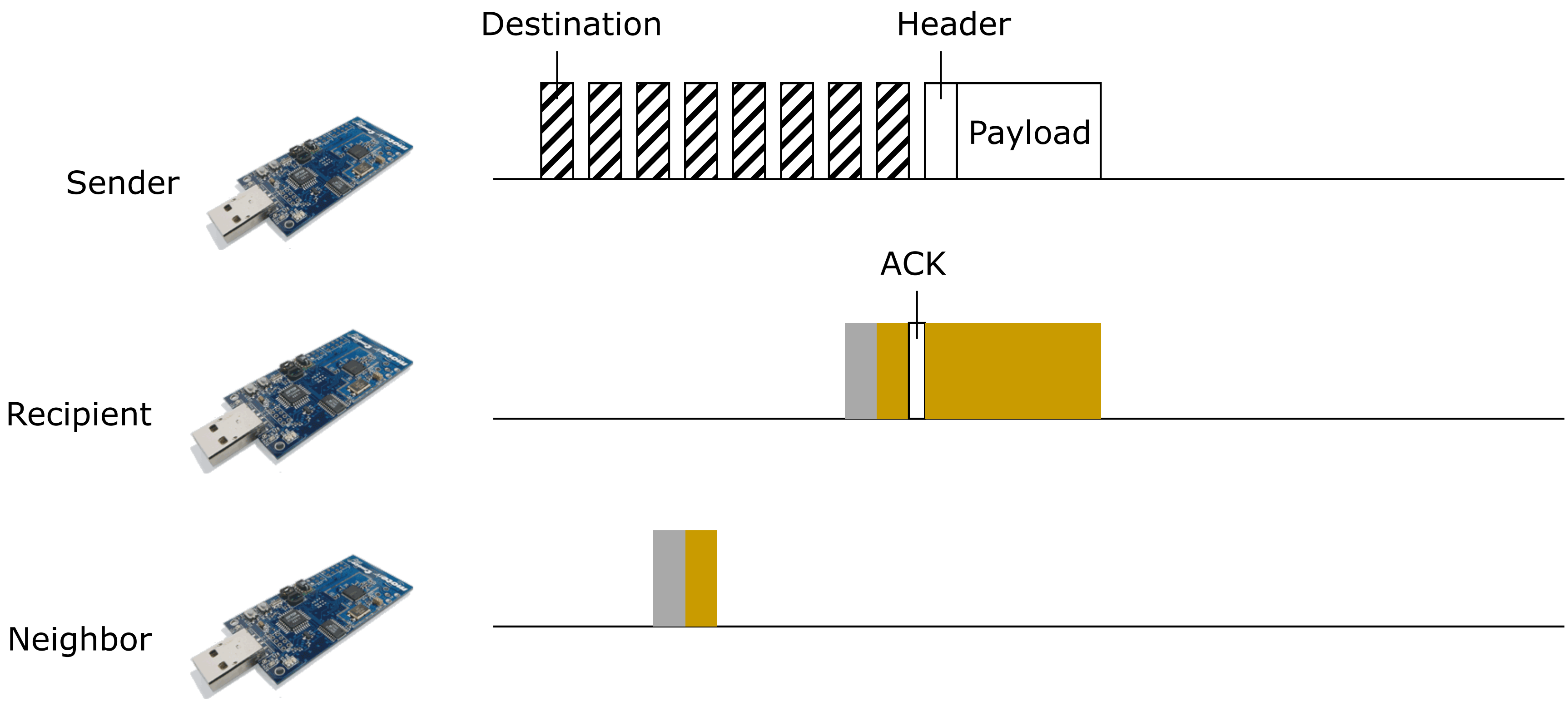
B-MAC: Room for Improvement



X-MAC: Overhearing Avoidance



X-MAC: Preamble ACKing

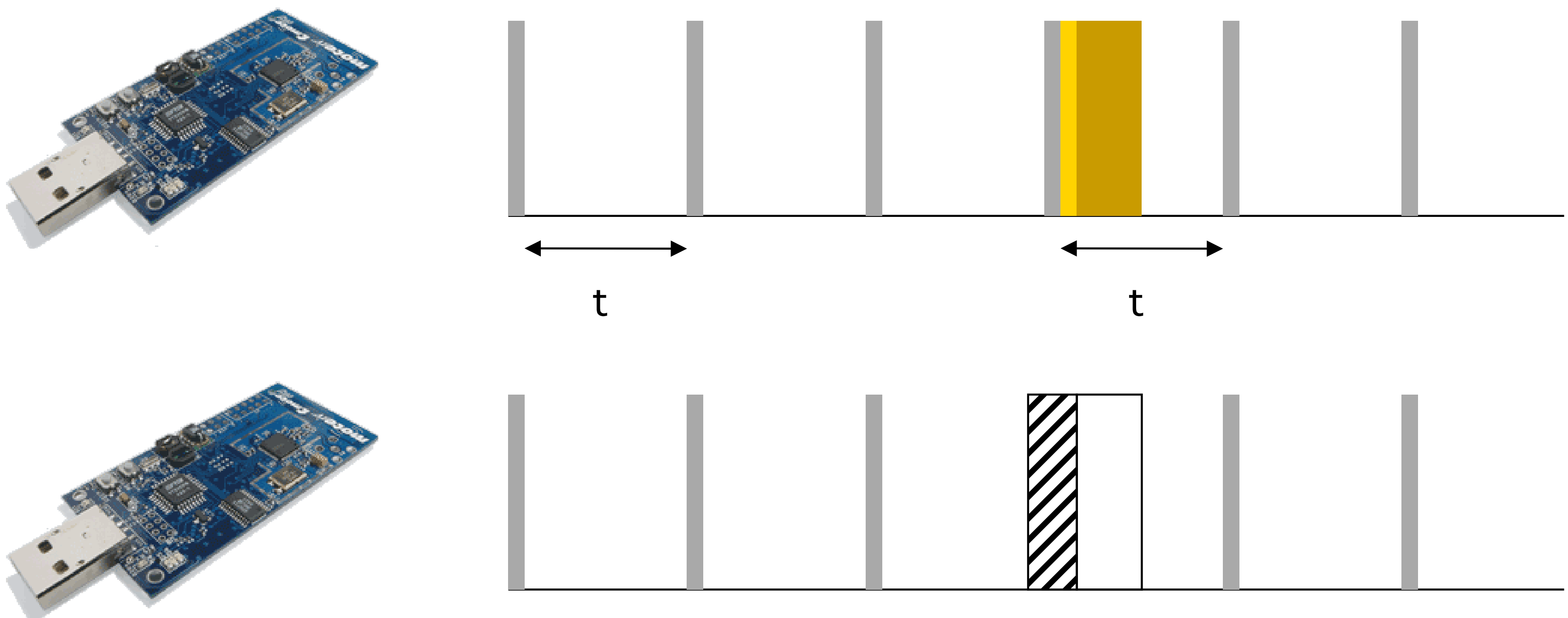


A Look at X-MAC

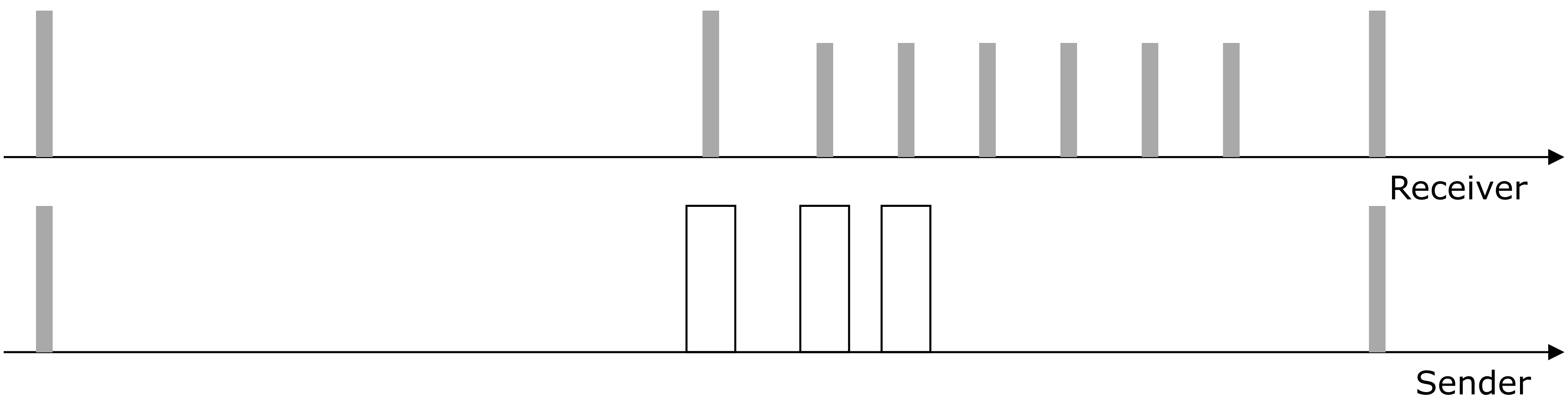
- Better latency, throughput, and power consumption than B-MAC
- Little energy consumed by overhearing
- Still simple to implement

- On average, cuts preamble by half → sending packets is still expensive

SCP: Scheduled Contention + Channel Polling



SCP: Adaptive Channel Polling



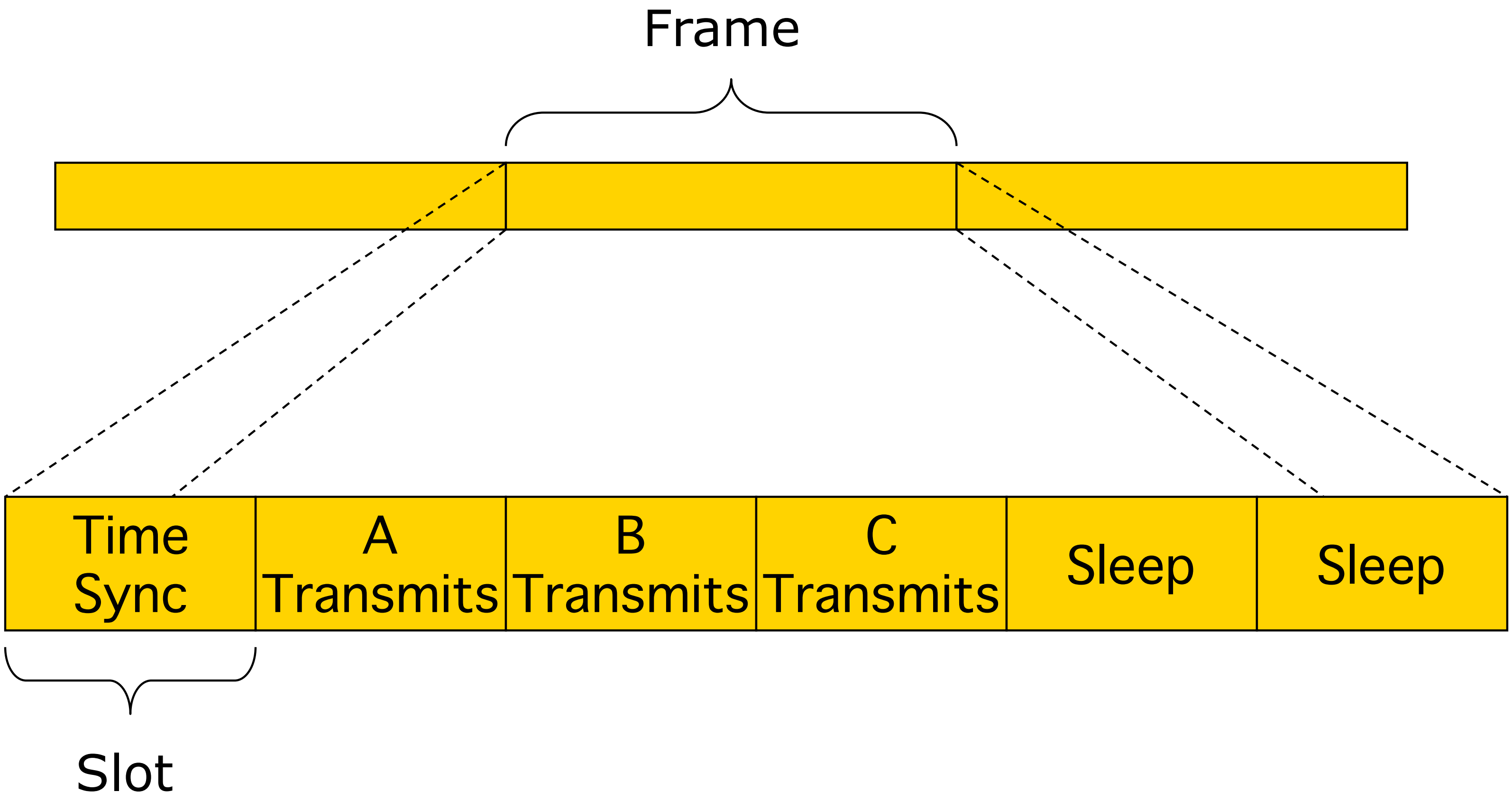
- Add N sub-intervals whenever packet received
  - Reduces latency of bursty data
- Continue adding sub-intervals as long as they're needed, and as long as there's room

A Look at SCP

- Channel polling: low overhead when idle
- Scheduled contention: low cost to send
- Low latency for multi-hop traffic

- Complex to implement
- Overhead due to time synchronization
  - Reduced by piggybacking on data packets

TDMA

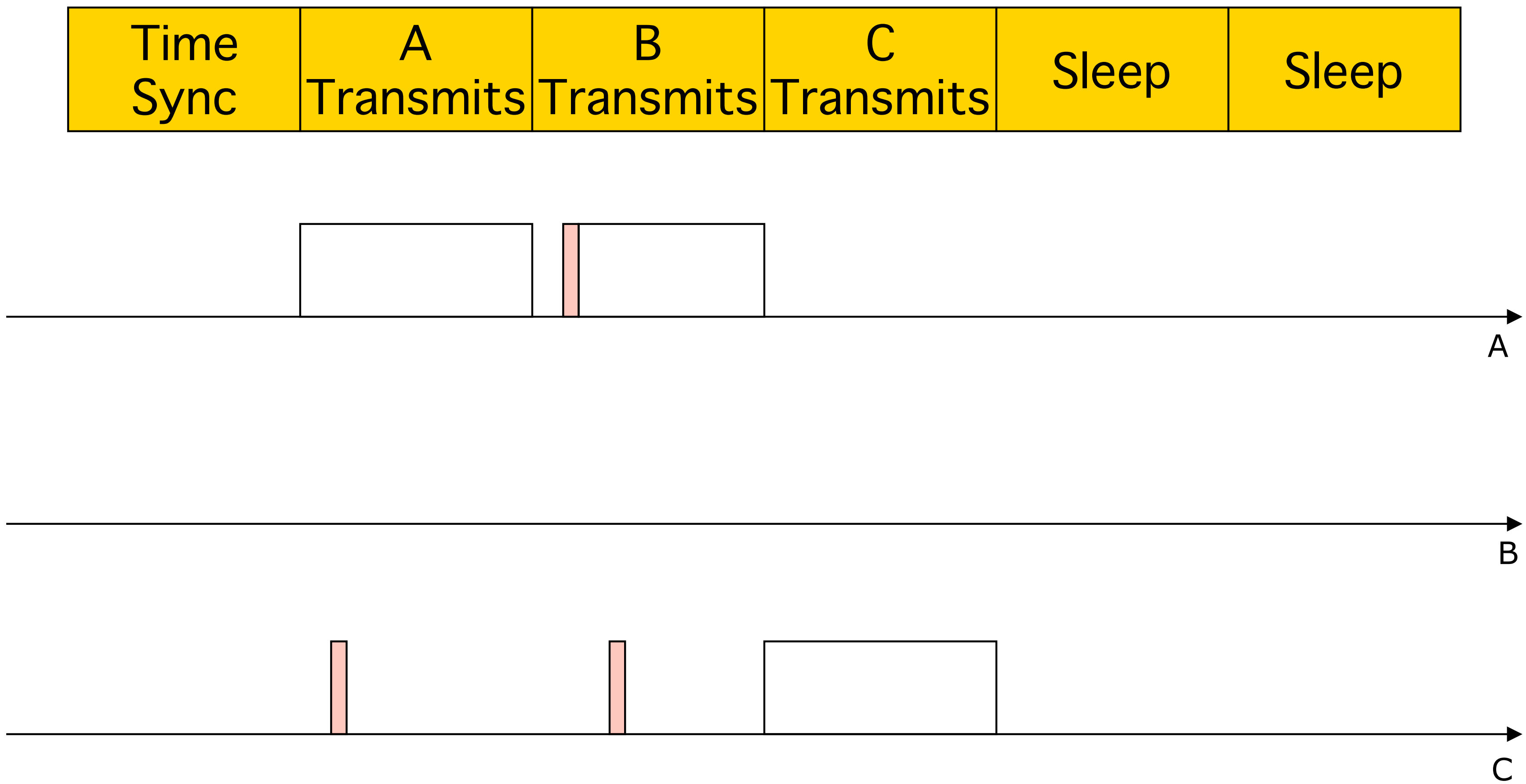

Frame
Time Sync
A Transmits
B Transmits
C Transmits
Sleep
Sleep
Slot

A Look at TDMA

- Predictable latency, throughput, and duty cycle
- Low packet loss due to less contention

- Schedules must be changed when nodes leave/enter neighborhood
- Time synchronization overhead
- Slots wasted when scheduled node has nothing to send
- Complex in multi-hop networks

Z-MAC: TDMA + Channel Polling



A Look at Z-MAC

- Reduces waste from unused slots
  - Higher throughput and lower latency
- Throughput, latency, and duty cycle no worse than pure TDMA

- Nodes still stay awake if no one transmits
- Still overhead from time sync

So Why Are They Rarely Used?

- MAC protocols have radio-dependent requirements beyond normal application code
  - Turn radio on/off, low latency I/O, carrier sense, etc.
- Typically implemented by forking radio stacks
  - Hard to implement
  - Must be maintained as original radio stack changes

- MAC layers supported by TinyOS today:
  - CC2420: BoX-MAC-2 (enhancements to X-MAC)
  - CC1100 and CC2500: Wake-on-Radio, X-MAC, B-MAC, BoX-MAC-1, BoX-MAC-2

A Unified Approach: MLA

- MAC Layer Architecture (MLA) [Klues 07]
  - Low-level abstractions of radio functionality
  - High-level implementations of common MAC logic

- Used to create 5 platform-independent MAC
  - B-MAC, X-MAC, SCP, TDMA, variant of Z-MAC

References

- [S-MAC] W. Ye, J. Heidemann and D. Estrin, Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks, IEEE/ACM Transactions on Networking, June 2004.
- [B-MAC] J. Polastre, J. Hill, and D. Culler, Versatile Low Power Media Access for Wireless Sensor Networks, ACM Conference on Embedded Networked Sensor Systems (SenSys'04), November 2004.
- [Z-MAC] I. Rhee, A. Warrier, M. Aia, and J. Min, Z-MAC: a Hybrid MAC for Wireless Sensor Networks, ACM Conference on Embedded Networked Sensor Systems (SenSys'05), November 2005.
- [X-MAC] M. Buettner, G. Yee, E. Anderson, R. Han, X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks, ACM Conference on Embedded Sensor Systems (SenSys'06), 2006.
- [SCP] W. Ye, F. Silva, and J. Heidemann, Ultra-Low Duty Cycle MAC with Scheduled Channel Polling, ACM Conference on Embedded Networked Sensor Systems (SenSys'06), November 2006.