Architecture

Master runs a state machine with the following states:

1. idle (entered upon reset)
2. waiting for user to hit a remote mole

Transitions are:

1. upon reset: randomly picks a mole slave node (excluding the previously picked one), sends a message of type 'arm', and transitions from idle state to 'waiting for user' state
2. in 'waiting for user' state: sends message of type 'request status' to the chosen node, upon getting 'status type' reply state remains unchanged, but upon getting a 'scored' type reply, state transitions from 'waiting for user' to idle state
3. upon timeout expiration master sends a 'reset' message to the mole node

Each slave runs a state machine with the follwing states:

1. idle (entered upon reset)
2. armed (mole is awake and awaiting to be hit by user)
3. scored (mole hit but the fact was not yet communicated to master)

The transitions are:

1. upon receiving 'arm' message, transition from idle state to armed state
2. upon detecting light intensity change in 'armed' state, transition to scored state
3. upon receiving status message in 'scored' state, reply with message of type 'stored' and transition to idle state
4. upon receiving status message in other states, reply with current state

Invariant: the slave never initiates communication: all its packates are replies to packets from the master node.
Our system does not implement acknowledgements for simplicity. Master ignores lost replies to its status requests, but re-issues them to the same node indefinitely.
Our system reduces timeouts based on the number of the round in order to challenge the player.

Latency Estimate

The latency comes from the following sources:

1. operating system and network stack overhead on sender node
2. MAC access delay (due to contention)
3. Flight of signal (negligible)
4. Delay on receiver due to **low-power listening** feature of BMAC

   The BMAC LPL protocol implies that the receiving node will only check for packets at a preset interval, which in our case is configured to be 100 ms in `bmac.h` as `BMAC_DEFAULT_CHECK_RATE`. We conclude that this source of delay is the most significant. So, until a remote mole is awake, one might have to wait more than 100ms.

Battery Lifetime Estimate

The master node polls the designated mole slave node (and only that node) at a fixed interval of 1 second. Each 'poll' is an exchange of exactly two packets (request packet and response packet). Each packet payload is below 4 bytes (node id and message type). Estimate header and checksum overhead at two bytes. Overall conservative estimate is 6 bytes. So, the data amount to transmit is about 6*2*8=96 bits per second, round to 128 bps. Radio bandwidth is 256 Kbps in burst rate. So, the data transmission can be performed in about 128/256k = 500 us, round to 1 ms. This means a duty cycle of 1 ms of transmission every second, i.e. 0.1%. Current during active transmission is 20 mA (see class slide on RF230), which at 100% duty cycle would drain two 2300 mAh AA batteries in 2300 * 2 mAh / 20 mA = 460 h. But, at 0.1% duty cycle this is 460 000 h. From this needs to be subtracted the idle power consumption (72uA rounded to 0.1uA), which would drain the battery in about 46,000 h. But even that is not the bottleneck. From this needs to be subtracted the compute power for the microcontroler (10mA). In our implementation we do not release the processor, therefore tihs will be consumed at 100% duty cycle. Therefore the battery will be drained faster than in 460 h = 19 days.