EMBS Assignment

Jake Coxon

December 1, 2012

0.1 Development

todo: how did i come to these design decisions The point of the source node is to transmit a packet during the reception phase of one of the three sink motes. For a given mote, any time a beacon is received with a payload n, there is nt time until the beginning of the reception phase and then $12t$ time until the beginning of the next sync phase. This can then be repeated indefinitely.

The first part of the program in this case is calculating t, which can be done by finding the difference in time between two separate beacons from a mote (t_1 and t_2). The beacons are not necessarily consecutive, so

$$t = \frac{t_2 - t_1}{n_1 - n_2}$$

$$\text{reception}_{\text{next}} = t_2 + n_2 t$$

$$\text{phase}_{\text{next}} = \text{reception}_{\text{next}} + 11t$$

If however n_1 is 1 then this is the last beacon of the synchronisation phase and so it is impossible to find t during this cycle. Fortunately it is known that the next sync phase is $12t$ time where $t \geq 500$ so the program should wait $12 \times 500\,\text{ms}$ and try again.

The next part of the problem comes when trying to calculating these values for three motes in parallel since all motes are on different channels.

The program tracks a state for all motes; initially 'waiting-to-sync'. The program will arbitrarily start with Sink A and wait for two beacons. Following this, the state is set to 'successfully-synced', the reception phase is calculated and a timer is set to fire at the reception phase.

If the program is waiting to sync and does not receive a beacon in 1500 ms it should switch to a new mote which has state 'waiting-to-sync'. This is because the program does not know if the mote is in synchronisation phase and therefore could be waiting for a beacon for up to $11 \times 1500\,\text{ms}$.

Once a packet is transmitted during the reception phase, it is known that the the next sync phase for this mote is at $11t$ time, so the timer can be started for this time and the whole process can be repeated. If this timer fires and the program is already trying to sync a mote then the new mote is 'queued' by setting the state to 'waiting-to-sync'.

Infact, the program knows whatever payload it receives will be n_{max} (The given n for the mote). Subsequently, the next reception phase can be calculated

$$\text{reception}_{i+1} = \text{reception}_i + (n_{\text{max}} + 11)t$$

This mean the program does not have to waste time going to the syncing phase every cycle (although we may like it to after some amount of time.)

The final problem to consider is when $n_{\text{max}} = 1$. This is a problem because there will not be two beacons in a single cycle and secondly the program does not know for sure that $n_{\text{max}} = 1$. This can be alleviated by waiting for the first beacon in the following cycle and calculating t. However, the program could potentially have to wait for 12000 ms[1] *without switching channels* else the program could miss a beacon where $n = 2$

Sink A		Sink B		Sink C		No. Packets
n	t ms	n	t ms	n	t ms	
10	500	4	700	5	1500	14
2	541	3	912	6	1101	11
6	1407	5	567	2	1207	12
2	725	8	868	4	1043	15
10	683	7	1308	2	685	14
1	683	7	1308	2	685	17
2	500	2	500	2	500	9
7	1308	1	1500	2	685	4
8	1500	1	1000	7	1000	7

Figure 1: Test data

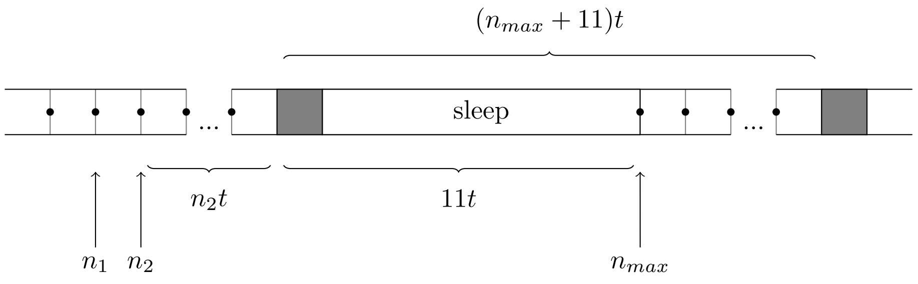

Figure 2: Timeline

[1]Because $500\,\text{ms} \leq n \leq 1500\,\text{ms}$, the next reception phase could be anywhere between $12 \times 1500\,\text{ms} = 6000\,\text{ms}$ and $12 \times 500\,\text{ms} = 18000\,\text{ms}$