Optimizing Timeout-Based Sleep Algorithms

Athina Markopoulou, Nick Bambos
Electrical Engineering Department, Stanford University
(amarko@stanford.edu)

***Abstract*— For wireless sensor networks that are expected to have a long lifetime while operating on limited energy resources, energy efficiency is a major concern [1]. In particular, it is well known that idle listening of the radio causes significant waste of energy; a power management technique [2] to deal with this problem is to switch to the lower energy consumption SLEEP mode during idle times. The effectiveness of this technique depends on (i) the duration of idle times (ii) the power consumption and performance characteristics of the sensor node and (iii) the sleep schedule/algorithm. A significant body of work focuses today on developing sleep schedules to meet energy and performance requirements. A class of simple, thus easy to implement, algorithms can be based on timeouts: nodes switch from ON to SLEEP, when the radio has been idle for longer than a Timeout period; clearly, the choice of Timeout becomes a critical parameter.**

In this extended abstract, we propose and analyze fixed timeout algorithms for a single sensor node, to minimize energy consumption. First, we consider the case that nothing is known about the duration of idle times; we choose a fixed timeout based on the power consumption characteristics of the node, and we prove that the energy consumed by this fixed timeout algorithm is at most twice the energy consumed by an offline algorithm. Second, we consider that the distribution of idle times is known and we show how to choose a fixed timeout value so as to minimize the average energy consumption; as a concrete example, we analyze the interesting case of Poisson packet arrivals. We are currently working on several extensions of this WIP, including (i) adaptive timeouts (ii) network-wide algorithms and (iii) performance metrics such as delay and connectivity.

I. Model and Problem statement

Fig.1 shows the simple state machine for the radio of a sensor node.

- *Modes.* The radio can be in one of two modes, ON or SLEEP (this can be generalized to several modes). When it is ON, it can forward packets but it spends P_{on} power in every time slot. When the node is in SLEEP mode, it spends negligible power (for simplicity, $P_{sleep} \simeq 0$, although this is not essential for the analysis) but it cannot forward packets.
- *Mode Transitions.* When the radio switches from SLEEP to ON, it spends wake-up energy E_{tr}. Energy is also spent on the transition from ON to SLEEP, but it is in general much less (we ignore this cost for simplicity, although this assumption is not essential for the analysis). Finally, we assume instantaneous transitions; in the next steps of this work we plan to consider transition delay D_{tr}.
- *Packet Arrivals and Forwarding.* Packets can arrive at the radio from two sources: either the sensor device at the node itself collects data, or the node acts as a relay forwarding packets from its neighbors. In either case, once a packet arrives, the radio must forward it immediately. If the radio is in SLEEP mode, it needs first to switch ON and then forward the packet.
- *Timeout.* Let t_i be the idle time since the last packet arrival. The role of the timeout algorithm is to observe for how long the node has been ON and idle and to switch the node from ON to SLEEP as soon as t_i reaches the timeout value T. The intelligence of a timeout algorithm lies in the choice and/or adaptation of T.

Fig.2 shows an example of state transitions as a result of packet arrivals and timeouts. Packets arrive at times $t_1, t_2, ..., t_5$. When packets $1, 2$ arrive, the radio is ON and immediately forwards them. After being idle for $s_1 - t_2 = T$ time, the radio switches to SLEEP. When packet 3 arrives, the radio wakes up immediately and serves the packet. It goes back to sleep at s_2, after it has been idle for $s_2 - t_5 = T$ time.

For a given idle period t_i, it may or may not worth switching from ON to SLEEP, depending on the operating (P_{on}) and wake-up (E_{tr}) costs. It worths switching if and only if the idle period is long enough to balance out the wake-up cost: $t_i P_{on} \geq E_{tr}$, i.e.

$$t_i \geq t_c = E_{tr}/P_{on} \quad (1)$$

where t_c a critical time that depends only on the energy consumption characteristics of the node. However, the actual energy savings from sleeping depend not only on t_c but also on the actual idle times.

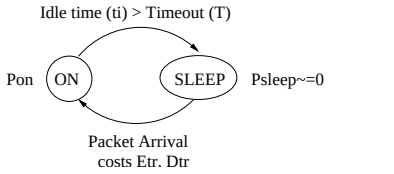

Fig. 1. State machine for the radio (states, transitions, associated costs and events).

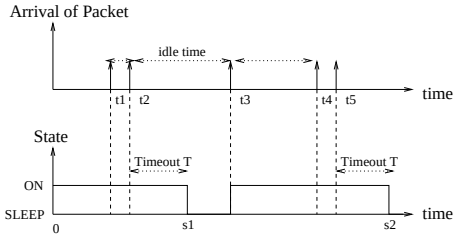

Fig. 2. Example of state transitions, depending on the arrivals of packets and on the timeout algorithm.

In a timeout-based algorithm, we have to make decisions online by comparing the idle time (since last packet arrival) to a timeout. Let us call A_T the algorithm that uses fixed timeout T: "*if the radio is idle for time T, then switch from ON to SLEEP*." T is the only parameter that needs to be optimized, based on the power consumption of the node (captured by t_c) and the packet arrival process (captured by t_i). In the next two sections, we choose and analyze timeout values T.

II. Choosing Timeouts for Unknown Arrival Process

First, we consider the case that nothing is known about the packet arrival process, or equivalently the t_i's. A_T is an online algorithm because it makes decisions when to go to SLEEP without knowing the t_i's in advance. An offline algorithm knows all idle times and can make the best choices; thus it is the benchmark for comparison.

In particular, at each packet arrival, the offline algorithm compares the next idle time to t_c and decides whether to switch or not immediately (at the beginning of the idle time). E.g. the idle period $t_3 - t_2$ was long enough and the offline algorithm would have switched to SLEEP immediately after serving the packet at t_2, instead of waiting until s_1 and wasting $P_{on}(s_1 - t_2)$ being idle. In practice, the idle times are not known to the online algorithm A_T before the next packet arrival.

Following a competitive analysis approach [3], we compare the energy E_{A_T} spent by the online algorithm A_T, to the energy $E_{offline}$ spent by the offline algorithm on the same sequence of t_i's. The following proposition is the building block for the rest of the analysis.
Proposition 1. An online algorithm that uses a fixed timeout $T = t_c$, thus called A_{t_c}, spends at most twice the energy that an offline

algorithm would have spent for the same sequence of t_i's.

Proof. Let t_i be the idle time since last packet arrival $t_0 = 0$.

At time t_0, the offline algorithm knows t_i and makes the optimal choice accordingly:

$$E_{offline} = \begin{cases} P_{on} t_i, & if\ t_i < t_c\ (no\ switch) \\ P_{on} t_i = E_{tr}, & if\ t_i \geq t_c\ (switch\ at\ t_0) \end{cases} \quad (2)$$

The online algorithm A_{t_c}, does not know t_i in advance, so it stays ON and only switches to SLEEP if t_i reaches the timeout $T = t_c$.

$$E_{t_c} = \begin{cases} P_{on} t_i, & if\ t_i < t_c\ (no\ switch) \\ P_{on} t_c + E_{tr} = 2E_{tr}, & if\ t_i \leq t_c\ (switch\ at\ t_c) \end{cases} \quad (3)$$

Therefore $E_{t_c} \leq 2E_{offline}, \ \forall t_i$. The same competitive ratio holds for every t_i, and therefore for the entire sequence of t_i's. Furthermore, this is the best competitive ratio achievable by any fixed timeout:

Proposition 2. No other online fixed timeout algorithm can achieve a competitive ratio smaller than 2.

Proof. For another timeout algorithm A_{T_x} with $T_x \neq t_c$, enumerate the cases for the order of t_i, t_c, T_x, and compare again the energy consumption for each resulting interval. We omit the details for lack of space. This general line of reasoning is inspired by [3].

Adaptive Timeouts. What if we adapt T with time as we observe arrivals and obtain an estimate for the next idle durations? There are many prediction techniques to estimate t_i^{next} based on its observed history. However, only simple techniques are implementable in a light-weight sensor node. Interestingly, there are very simple prediction techniques which are analyzable within this competitive analysis framework, similarly to [3], but are omitted from this WIP for lack of space. E.g. predict $t_i^{next} = t_i^{previous}$ and adapt $T = t_c$ if $t_i^{next} > t_c$ and $T = 0$ otherwise; this can provably achieve a competitive ratio of 3.

III. Choosing Timeouts for a Known Arrival Process

In some cases, it may be possible to know the distribution $f(t_i)$ of packet inter-arrival (or equivalently idle) times t_i. For example, the sensor may collect data periodically or receive packets from the neighbors according to some rule. Knowing $f(.)$ enables us to tune T for it and also carry out a probabilistic as opposed to a worst case analysis. In this section, we choose a fixed timeout T to minimize the average energy spent on a sequence of t_i's drawn from $f(t_i)$.

As discussed in the previous section, for a given idle time t_i the timeout algorithm A_T spends energy:

$$E_T = \begin{cases} P_{on} t_i, & if\ t_i < T\ (do\ not\ switch) \\ P_{on} t_i + E_{tr} = P_{on}(t_i + t_c), & if\ t_i \geq T\ (switch\ at\ T) \end{cases} \quad (4)$$

E_T is known as a function of t_i, t_c, P_{on}, if T is known. Because t_i is now a random variable drawn from distribution $f(t_i)$, we can compute the average energy $E[E_T]$ spent on a sequence of idle times, by averaging over t_i; then we can choose T to minimize $E[E_T]$.

Poisson Arrivals. As a concrete example of calculation, let us consider Poisson arrivals. This is also an interesting process in itself because it can arise as a superposition (e.g. a large number of neighbors sending their packets through the node). The inter-arrival times t_i are now exponentially distributed, say with parameter λ: $f(t_i) = \lambda exp(-\lambda t_i)$ for $t_i \geq 0$. The expected value of the energy spent by A_T can be easily computed:

$$\begin{aligned} E[E_T] &= \int_0^\infty E_T f(t_i) dt_i \\ &= \int_0^T P_{on} t_i \lambda e^{-\lambda t_i} dt_i + \int_0^T P_{on}(t_i + t_c) \lambda e^{-\lambda t_i} dt_i \quad (5) \\ &= \cdots = \underbrace{\frac{P_{on}}{\lambda}(1 - e^{-\lambda T})}_{} + \underbrace{P_{on} t_c e^{-\lambda T}}_{} \end{aligned}$$

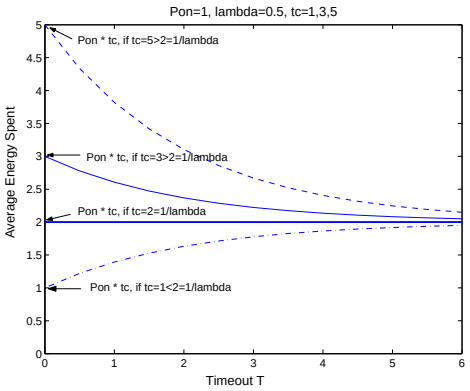

Fig. 3. Average energy spent $E[E_T]$ on a sequence of Poisson packet arrivals, as a function of the timeout T, as computed in Eq.(5). The shape of $E[E_T]$ depends on the arrival rate λ, the critical value t_c and the difference $t_c - 1/\lambda$. As a numerical example, we used: $P_{on} = 1$, $\lambda = 0.5$ and $t_c = 2$.

Let us briefly discuss the intuition and tradeoffs behind this formula. The average idle time is $1/\lambda$; the critical time is t_c and recall that $P_{on} t_c = E_{tr}$. The first term in Eq.(5) is the energy spent while ON P_{on}/λ, provided that we are ON: $Pr(t_i > T) = 1 - e^{-\lambda T}$. The second term is the energy spent waking up $P_{on} t_c = E_{tr}$, provided that we were in SLEEP mode: $Pr(t_i < T) = e^{-\lambda T}$.

In Fig. 3, we plot $E[E_T]$ as a function of the timeout T. It turns out that $E[E_T]$ is a monotone function of T (which can be explained by the memoryless property):

- It is increasing in T if $tc < 1/\lambda$ (i.e. idle times are on average longer than the critical t_c). To save the most energy, we should go to SLEEP immediately after serving a packet ($T \longrightarrow 0$).
- It is decreasing in T if $tc > 1/\lambda$ (i.e. idle times are on average shorter than t_c). To save the most energy, we should be hesitant to go to SLEEP and at the extreme always stay ON ($T \longrightarrow \infty$).
- It is constant if $tc = 1/\lambda$. Then we spend on average $P_{on}/\lambda \ \forall T$.

Similar analysis can be applied to optimize T for other arrival processes. E.g. we tried it for uniform distribution, and we observed that, unlike Poisson, $E[E_T]$ has a single maximum in T. Finally, in a tandem or in a network of nodes, our analysis can be used to tune the timeout at each node depending on its load (which now is the superposition of its own sensed data and the traffic forwarded by neighbors).

IV. Extensions and Ongoing Work.

In addition to the extensions mentioned above (adaptive timeouts, general arrival processes, network-wide algorithms), we are currently working on including performance - in addition to energy- objectives. For example, when the wake-up delay is non-negligible, there is a performance penalty every time a packet arrives the radio wakes up. We are currently working on optimizing timeouts for both energy [4] and performance (such as delay [5] and connectivity [6]) guarantees.

References

[1] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware wireless sensor networks", *IEEE Signal Processing Magazine* 19(2), pp.40-50, March 2002.

[2] L.Benini, A.Bogliolo, G.De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management", *in IEEE Trans. on VLSI Systems*, Vol.8, No.3, June 2000.

[3] A. Karlin, M.Manasse, L.McGeoch, S.Owiski, "Competitive Randomized Algorithms for Non-Uniform Problems", *Algorithmica*, pp.543-571, 1994.

[4] A. Sinha and A. P. Chandrakasan, "Operating System and Algorithmic Techniques for Energy Scalable Wireless Sensor Networks," *in Proc. of the 2nd Intl Conference on Mobile Data Management (ICMDM*, Jan. 2001.

[5] C.Schurgers, V.Tsiatsis, S.Ganeriwal, M.Srivastava, "Optimizing sensor networks in the energy-latency-density design space," IEEE Trans. on Mobile Computing 1, 2002.

[6] A.Goel, S. Rai, and B.Krishnamachari, "Sharp thresholds for monotone properties in random geometric graphs," *in Proc. ACM STOC*, June 2004.