Distributed Power-Management Techniques for Wireless Network Video Systems

Nicholas H. Zamora, Jung-Chun Kao, Radu Marculescu
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
e-mail: {nhz, jungchuk, radum}@ece.cmu.edu

Abstract

Wireless sensor networks operating on limited energy resources need to be power efficient to extend the system lifetime. This is especially challenging for video sensor networks due to the large volumes of data they need to process in short periods of time. Towards this end, this paper proposes two coordinated power management policies for video sensor networks. These policies are scalable as the system grows and flexible to video parameters and network characteristics. In addition to simulation results, our prototype demonstrates the feasibility of implementing these policies. Finally, the analytical framework we provide gives an upper bound for the achievable sleep fraction and insight into how adjusting select parameters will affect the performance of the power management policies.

1. Introduction

Shrinking technology sizes and CMOS image sensors [1] make coordinated, wireless, battery-powered video networks viable. Wireless sensor networks operating on limited energy resources [2] need to be power efficient to extend system lifetime and reduce maintenance costs with fewer battery replacements. This has to happen at each level of design abstraction, from circuit level [3] up to software and network levels [4], especially for video networks as image sensors (cameras) tend to have much higher power consumption rates compared to other sensors.

This work focuses on the problem of coordinated power management (PM) for video sensor networks working at the *application level*. More precisely, we propose two coordinated techniques for video sensor PM which are affordable using today's technology. The first technique involves extending the idea of local timeouts to a set of nodes, while the second technique uses a voting algorithm to determine when nodes should be shut down. These techniques do *not* require location or topology awareness. The techniques we propose are also scalable as the number of cameras in the system increases; this is because only information from the immediate neighbors is used in the PM decisions (i.e. – no remote message forwarding or cooperation is necessary to use these techniques). Therefore, these techniques are flexible, handling various network configurations successfully.

2. Prior work and novel contribution

Dynamic PM policies have already been studied [5] and coordinated PM policies for sensor networks are beginning to appear [6]. The authors in [6] show a policy which can guarantee a minimum coverage while maintaining the network connectivity. However, the assumption of circular sensing areas and the sparse nature of the coverage which often occurs in video surveillance systems make the technique inappropriate for the systems we envision. When there is dense video coverage, the sensors' fields of view (FOVs) overlap and the authors in [7] attempt to save energy by partitioning the data to be captured. This brings to light a whole class of applications which rely on multiple video sensors such as in Fig.1A, 1C, and 1D. For this class of systems, it is desirable that a coordinated PM policy be sensitive to the node interdependencies inherent in the application. The PM policies we propose in this work operate synergistically with such applications; we later show how this is possible.

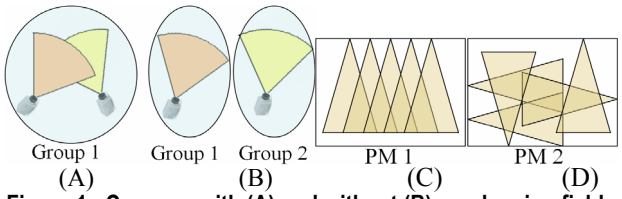


Figure 1 - Cameras with (A) and without (B) overlapping fields of view (FOVs). In (C), PM 1 could be a custom policy tailored to a particular coverage type. Our policies are compatible, and perform well with, both hand-crafted systems, as well as randomly distributed networks (as in PM 2 in (D)).

The authors in [8] use a hybrid automaton model which considers the underlying application when executing the PM policy. If the architecture contains multiple power modes with varying degrees of functionality, then a technique such as the one described in [9] can utilize these modes to minimize the energy consumption based on prediction models for the environment.

Multiple camera systems research is beginning to appear [10][11], but real-time multiple video streaming over a mesh network will require improvements in current technology. Protocols such as 802.11n [12] and energy devices such as ultra-capacitors [13] could enable these systems in the next decade or even later.

The policies we propose are unique for several reasons:

- First, most of the current research assumes the power consumed for wireless transmission dominates and the power rates required to process the video data locally are relatively small. Our proposed PM policies have a low energy overhead, while still benefiting from information from the neighboring nodes. The required communication to utilize these policies is not nearly as expensive as real-time video streaming. We show that the overhead for our PM policies is indeed negligible.

978-3-9810801-2-4/DATE07 © 2007 EDAA

- Second, the policies presented in this paper consider the content of the video data sensed both locally and by other video nodes within the network. In other words, the moving macroblocks sampled by neighboring nodes *are* considered when each node executes its own PM policy.
- Third, the proposed policies use a best-effort approach to utilizing all the information available and without requiring special hardware like GPS receivers or running additional topology or location-discovery algorithms. Our policies perform well in both hand-crafted, as well as randomly distributed network systems, as in Fig.1C and 1D.
- Finally, these PM policies are completely and inherently scalable with the size of the system because they only consider the neighbors one hop away from the local node. For very dense networks, these policies can be restricted to only listen to a maximum number of immediate neighbors.

The paper is organized as follows. The next section describes the new PM techniques. Section 4 describes the experimental setup for validating the proposed policies, while Section 5 discusses in detail the results of those experiments. Section 6 presents an analysis which predicts the achievable sleep time. Finally, Section 7 summarizes our findings and points out future directions of research.

3. New power management policies

3.1 A cost-based function for PM policies

For the newly proposed PM policies, the decisions made are *when* to put a node into a power-down mode in order to maximize the power savings without sacrificing the system performance. Our novel cost metric *(Cost)*, which quantifies the performance of a PM policy, combines the image misses and node inefficiencies as shown below:

$$Cost = \alpha \times T_{ineff} \times \frac{T_{total}}{T_{total} - T_{activity}} + \beta \times T_{miss} \times \frac{T_{total}}{T_{activity}} \quad (1)$$

where T_{ineff} is the number of inefficient timesteps, or timesteps in which the node is active but there is no interesting activity being sensed (i.e. – the node should be powered down ideally), T_{miss} is the number of timesteps the node is powered down and is missing interesting activity which it would sense if it were active, $T_{activity}$ is the number of timesteps in which there is interesting activity for the node to potentially sense, and T_{total} is the total number of timesteps in the experiment. Most previous cost metrics do not consider the sensed data [14][15], and so this cost metric is novel for video sensor networks. The positive weights α and β balance the importance of each error type. Eq.(1) emphasizes power consumption when $\alpha > \beta$; we show examples adjusting these parameters later in Section 5.

In eq.(1), each miss is more costly the less activity there is and, similarly, each inefficient time step adds more cost the more activity there is. If all time steps contain interesting activity (i.e. – $T_{activity} = T_{total}$), then this equation is not well defined. The ideal PM policy in that case keeps all nodes active all the time (i.e. – no PM); we include this policy later in our experiments for comparing our results.

3.2 Coordinated PM policies

We need to find an optimal technique which minimizes the cost in eq.(1). To find such a policy, we need *a priori* knowledge of the video data itself. Since we do not know the video data in advance, we cannot use eq.(1) to propose a new policy; instead we can use it to assess the new policies.

Our first proposed technique is to extend the local timeout policy to a distributed network of nodes. The well-known local PM policy [16] keeps a timer (for each node) indicating how long no movement has been detected and goes to sleep after this timer times out. In the coordinated version, when a timeout occurs, the node begins broadcasting a "timeout" message periodically. Each node powers down if it and all its network neighbors have timed out. The node then returns to active mode after a fixed sleep time has elapsed.

The second technique implements a *voting* algorithm to determine when to power down nodes. Periodically, each node considers its own movement information, as well as the most recent movement information from each neighbor; it powers itself down if fewer than $Votes_{allowed}$ nodes are detecting movement. Formally:

$$Votes_{movement} \le Votes_{allowed}\,(total\ votes) \quad (2)$$

where $Votes_{movement}$ is the number of nodes which are claiming there is currently movement detected, and $Votes_{allowed}$ is a threshold and a function of the $total\ votes$ collected. When eq.(2) holds, the node is sent to sleep for a duration determined by the fixed sleep time value. For this work, we use a static table lookup to implement this policy, while the more advanced techniques (e.g. weight each vote differently based on its correlation to the local node and then use a learning period to determine these correlations at system startup) are left for future work.

3.3 Considering inter-node dependencies

If the underlying video application combines data from multiple sensors simultaneously using data fusion [11], then the PM policies can be adjusted to work seamlessly with such applications. If the application is gracefully degrading, then our previous coordinating PM policies are applicable. Otherwise, the application may require the nodes be either fully active or asleep at any point in time in order to ensure an acceptable level of performance.

For non-gracefully degrading systems, the nodes form groups as shown in Fig.1A and 1B. Each group consists of the fewest video nodes which depend only on other nodes in the same group. Group communication problems are described in [17] and more recently in [18]. Note that existing group algorithms can be seamlessly integrated with the newly proposed coordinated PM policies. For instance, a *group leader* can be elected; this leader decides for each group member whether or not to transition into the sleep mode (Fig.2). The group leader then broadcasts sleep commands to the rest of the members in the group.

System Reset/Startup: i, j, time<=0
Discover nodes present, form groups, i++ (1)
Each node broadcasts leader score, $C_t^{\delta} \times Q_i^{\gamma}$ (2)
where C_t is current battery fraction remaining, Q_i is quality of image detected, in the range of 0 to 1
Each node elects a leader based on highest score, j++
while (true) (3)
Execute video application and coordinated PM policy for next frame
if time > i × T_{group} goto (1)
if time > j × T_{leader} goto (2)

Figure 2 - Coordinated PM policy for non-gracefully degrading applications

To select the group leader, each node should maintain a running (i.e. on-line) average of both the remaining battery capacity (C_t) and quality of the image (Q_i) detected. Selecting the group leader involves comparing these values and electing the node with both ample remaining battery, as well as a good image quality. Periodically, the groups are broken down and reconstructed to allow nodes to enter or leave the system; this is shown in Fig.2.

T_{leader} is the time a leader is assigned that position, and T_{group} is the amount of time the groups will stay constant regardless of nodes entering or leaving the system. When nodes enter, they must wait for group re-construction to join the network. However, when group/cluster leaders are not utilized, nodes entering the network are immediately recognized by their neighbors. A reasonable choice of these values is $T_{group} = m \times T_{leader}, m \geq 1, m \in Int^{+}$ (i.e., T_{group} is a positive multiple of T_{leader}). For selecting the group leader, the parameters δ and γ prioritize the importance of remaining battery and image quality, respectively, by serving as exponents in the cost metric (line (2)). Setting γ = 0 selects the node with the most remaining battery capacity as leader.

4. Simulation, prototype, driver application

To demonstrate the effectiveness of these PM policies, we have simulated and prototyped them using a complex pursuer-evader application. The prototype allows us to show that resource-constrained platforms are capable of running complex video applications. The simulations allow us to extend the experiments beyond what is possible to achieve using the prototype alone.

4.1 Accurate and flexible PM policy simulator

In addition to a first generation prototype, we have built an event-triggered simulator to test our proposed techniques. Using this simulator, we generate random configurations and human environments with adjustable parameters. Fig.3A shows a snapshot of a randomly-generated configuration consisting of 100 rooms, 50 people, and 50 cameras.

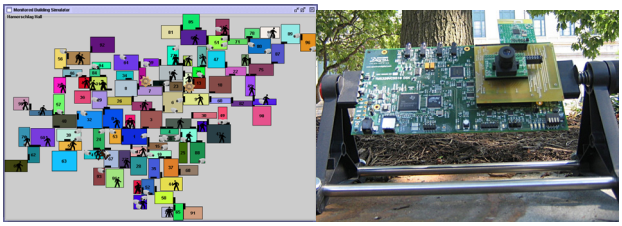
In the simulator, we can simulate any PM scheme in action. Additionally, we have implemented a social network where simulated people have friends, close friends, and even enemies, and they will tend to follow their friends around the building while avoiding their enemies. We have also implemented a collision detection and physics system for movement, ensuring that people contain an appropriate amount of inertia to model real-world movement.

Each camera has an arc-shaped FOV (see Fig.1B) where it can detect movement, and a wireless transmission power which limits the neighbors to which it can directly communicate. Our simulator assumes all broadcasts across the network experience a fixed end-to-end delay of 100ms. Due to the periodic nature of video capturing and processing, this fixed delay is a reasonable assumption which simplifies our simulations considerably. When network messages have variable delays, and even some nodes experience network starvation, our PM techniques will still succeed using the information from active neighboring nodes.

Due to its generality, the driver application we choose to test the effectiveness of the proposed PM policies is the pursuer-evader game. More precisely, we have a smart pursuer who receives motion information from the network to help find an escaping evader. The setup also includes a random pursuer; that is, a pursuer with no camera information who just searches the building at random. The other people in the building are innocent bystanders not involved in the game, but going about their own business. The smart pursuer knows the layout of the building and how to get from any room to any other (Fig.3A). The information the smart pursuer receives from the cameras includes when the evader is detected by any of the cameras. We also implement a false hit detection of 1%; that is, each time an innocent bystander moves in front of a camera, then that camera will report it as being the evader with that probability. The game begins with all people randomly placed throughout the building and ends when both the smart pursuer and the random pursuer catch the evader. Detailed simulation results are presented in Section 5.

4.2 DSP prototype

We have implemented a prototype which uses the TI 5510 DSP with an Omnivision CMOS VGA B/W camera and the Chipcon 2420 Zigbee transceiver (see Fig.3B).



(A) (B)

Figure 3 – Wireless video network simulator GUI, 100 rooms (A) and a prototype node using a TI DSP, an Omnivision CMOS camera, and a Chipcon Zigbee wireless transceiver (B)

We built four prototype nodes and have formed a multi-hop wireless network, using the intermediate nodes to packet hop video images from source to destination. We use two packet formats to send data between individual nodes; the packets sent over the RF channel are shown in Fig.4.

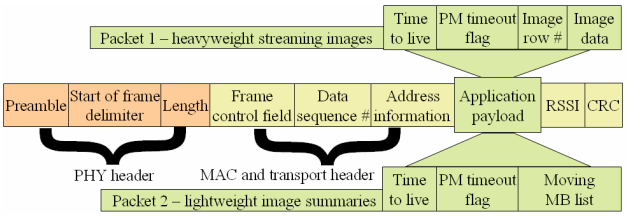


Figure 4 - Two packet formats to selectively choose when to stream real-time video and when to send lightweight video "summaries" of detected movement in the current frame

Packet type 1 shows the *image streaming* data where the packet contains the image data itself. Due to the packet length limit of 128 bytes, each image is broken up and sent across multiple packets (frames). The bottleneck in this case comes from the number of concurrent images which need to be transferred wirelessly due to bandwidth restrictions. The transceiver can achieve 250kbps, while the throughput for *image streaming* is 2-3 images per second per channel, depending on the channel characteristics.

For packet type 2, the *image summary* packet (i.e. – the list of macroblocks (MBs) which contain movement) might contain as many as half the MBs in the video frame is sent. If more than half the MBs contain movement, we list instead the MBs which do *not* contain any movement. For VGA frames of size 640x480, there are, at most, 600 MBs which

need listing; each MB takes 1 bit to indicate movement or no movement. For the *image summary*, the maximum size is 31 header bytes and 76 application payload bytes.

For each packet type, the proposed PM policies require a single timeout bit per image frame. This constitutes an overhead of 0.4% for the smallest size packet type 2, 0.1% for packet type 1, and so we can say with confidence that our power management policies present a very little overhead.

For each packet type, the time-to-live (TTL) portion of the application payload (see Fig.4) determines how many intermediate nodes the packet should be forwarded through before being discarded. This TTL value is application specific and is part of the application payload itself.

Our prototype nodes use carrier sense multiple access (CSMA) to avoid collisions and a cyclic redundancy check (CRC) to discard the corrupted packets. Also, we use random times for transmit retries to prevent nodes starvation.

	Packet type 1	Packet type 2
Avg. Latency	8.0 msec.	3.2 msec.
Avg. Energy Consumption	3.2 mJ	1.3 mJ

Table 1 - Time and energy required for transmitting wireless packets for each hop

In Table 1, we give the measured and calculated average latency and energy consumption values for the prototype system based on four video nodes. The latency values were acquired by using reported timestamps in the software. The energy consumption values were obtained by first measuring the power draw for the entire prototype using a digital multimeter, then integrating those power draw measurements over the average latency of the transfer. The entire prototype uses just over 400mW, while operating in the most power-hungry mode and transmitting data over the wireless channel.

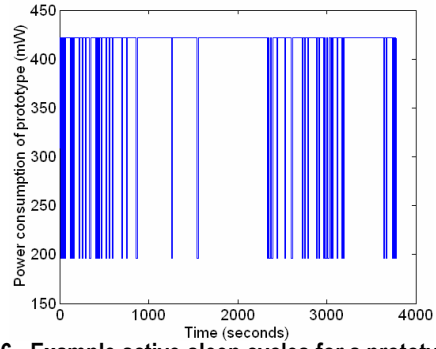


Figure 6 - Example active-sleep cycles for a prototype node using the coordinated timeout policy

Power measurements taken at one node during a typical experiment are shown in Fig.6. The video nodes in this experiment were aimed at different angles overlooking a busy parking lot on the CMU campus. The periods of high power consumption in Fig.6 correspond to high activity in the parking lot, and the low power consumption rates were achieved when the node experienced a coordinated timeout.

5. Experimental results

We simulated 72000 random building experiments in total, 1500 for each of the four PM techniques (2 of which are newly proposed here) and 12 different camera density setups; we compare performance results in Fig 5A.

The first technique we use for comparison is called *local* PM, and is the well known technique where each node keeps a timer indicating how long no movement has been detected and goes to sleep after this timer times out (e.g. 15 seconds in our experiments). In the local PM technique, the cameras sleep for a fixed period of time, 30 seconds, and wake back up after this time has elapsed. The important thing to note in Fig 5A is that the smart pursuer catches the evader faster and faster as the camera density increases. The flavor of PM used does not affect the catch speed considerably, which implies that, even though cameras do miss movement when they are put to sleep, no person goes undetected for a considerable length of time. This can be surprising when considering that the PM policies each send the nodes to sleep more than 50% of the time. The resulting power consumption for the local policy, coordinated timeout, and the voting policy average 36%, 19%, and 27% of the original rates, respectively. As such, the coordinated policies outperform the local policy.

There are additional system parameters which affect the performance of our PM policies. In Table 2, we list various system parameters and show their effects on our PM policies. Our PM policies respond to increased node density or increased FOVs by decreasing the sleep time achieved and improving the QoS of the application.

There is no effect on the PM policies by increasing the network communication, as long as all the nodes change their data rates identically. However, the network congestion may have an effect. Our policies are scalable with the number of nodes (and neighbors), so the network congestion will cause packets to be delayed or lost. When all PM flags fail to be transmitted or when a node becomes disconnected, our policies revert back to the local timeout policy and so will still perform just as well as the local policy.

The variation of the overall cost, calculated according to eq.(1) with α and β both equal to 1, is shown in Fig.5B. The voting PM policy performs the worst except for no PM policy because it is only relies on a snapshot of object detection information instead of a more reliable window of information over time. Although the local PM policy

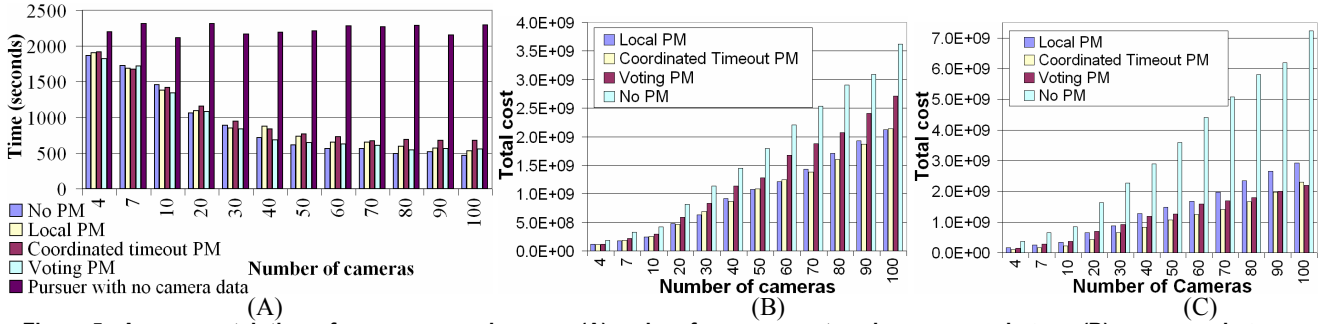


Figure 5 - Average catch times for pursuer-evader game (A) and performance costs using $\alpha = 1$ and $\beta = 1$ (B), $\alpha = 2$ and $\beta = 0.5$ (C) in a 100-room building using various PM techniques.

System Parameter	Case 1	Case 2	Effect on Coordinated Timeout Policy	Effect on Voting Policy
Building Variable (Nodes / m2)	0.1	0.2	From 0.1 to 0.2, cameras sleep 40% less and application improves 20%	From 0.1 to 0.2, cameras sleep 5% less and application improves 10%
Network Variable (Video Qual., kbps)	70	600	No effect if all nodes change	No effect if all nodes change
Camera Variable (FOV, degrees)	25	45	From 25 to 45, cameras sleep 25% less and application improves 10%	From 25 to 45, cameras sleep 20% less and application improves 10%

Table 2 - Each row shows a system parameter. We show the effect each parameter has on our PM policies assuming other variables remain constant in each row. Each data point is based on 1500 random wireless video network simulation runs.

performs well, the coordinated timeout policy in almost every case wins in terms of lowest cost.

To conserve the battery life even more, we set α=2, β=0.5, as in Fig.5C. The coordinated timeout policy is then most cost effective when there is low node density and the voting policy is most cost effective when there is high node density. With higher camera densities, the coordinated timeout policy should use shorter timeouts to maintain greater power savings while still maintaining low performance costs.

6. Analytical performance estimation

Next, we present an approximate analysis of these policies and derive an *upper bound* on the sleep time achievable. Additionally, we analyze *how closely* can we approach that bound. The analytical results show that our timeout and sleep time values are below a critical threshold and can be increased to reduce wakeup/shutdown cycles. Future work includes improving the accuracy of this model.

6.1 Coordinated timeout with predictive wakeup

One way to predict how often a node will be powered down is to discretize time into equal timesteps as in Fig.7.

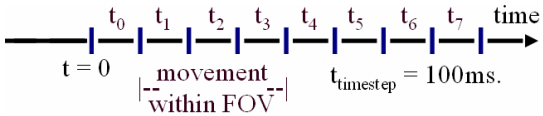


Figure 7 - Discrete timesteps at a video node with potential movement to be detected during timesteps t_1-t_3

For the *coordinated* timeout policy, the *expected time* until a node shuts down is:

$$E[t_{sd}(m_{to})|n] = \sum_{k=0}^{n} E[t_{sd}(m_{to})|\ k\ awake\ ] \times Pr(k\ awake\ |\ n) \quad (3)$$

$$= \sum_{k=0}^{n} E[t_{sd}((k+1)\times m_{to})] \times Pr(k\ awake\ |\ n)$$

where n is the number of neighbors, *to* is shorthand for "timeout", m_{to} is the number of timesteps required for a timeout, $E[t_{sd}(m_{to})]$ is the expected number of timesteps until a node enters into a low-power mode, and $k\ awake$ means k out of these n neighbors are currently active. In other words, we must consider the number of fully active nodes which are immediate *neighbors* to the node of interest. These active nodes are the ones considered when executing the PM policy. The term $E[t_{sd}((k+1)\times m_{to})]$ takes the form:

$$E[t_{sd}(z)] = t_{ts} \times E[timesteps\ \ until\ shutdown] \quad (4)$$

$$= t_{ts} \times \left(\begin{array}{l} (1-p_m)^z \times z + p_m \times (1-p_m)^z \times \\ \sum_{j=0}^{\infty} (Pr(no\ \ to\ \ in\ j\ \ timesteps) \times (m_{to}+j+1)) \end{array} \right), z = (k+1)\times m_{to}$$

where t_{ts} is the time for one timestep, p_m is the probability of movement in each timestep, and $Pr(no\ to\ in\ j\ timesteps)$ is the probability of not having a timeout in j consecutive timesteps. $E[timesteps\ until\ shutdown]$ can be determined as follows: One possibility is that the node and its neighbors experience an *immediate timeout* (i.e. – no movement detected in the first z timesteps). This occurs with probability $(1-p_m)^z$ and results in m_{to} timesteps until shutdown. Otherwise, for some j timesteps before the timeout begins, there can either be movement or no movement detected in each of those timesteps for each neighbor. However, there can not be a run of length m_{to} timesteps of no movement for all neighbors during this period or else that run would itself result in a timeout! The $(j+1)^{th}$ timestep contains movement, and the next m_{to} timesteps will all result in no movement detected at all neighbors. The resulting number of timesteps until shutdown is $m_{to}+j+1$ and the probability of this event occurring is $p_m \times (1-p_m)^z \times Pr(no\ to\ in\ j\ timesteps)$. This leads to eq.(4), which is depicted graphically in Fig.8.

The $Pr(no\ to\ in\ j\ timesteps)$ is the same probability as having no m_{to} heads in a row occur when flipping a weighted, two-sided coin j times, and is solved in [19].

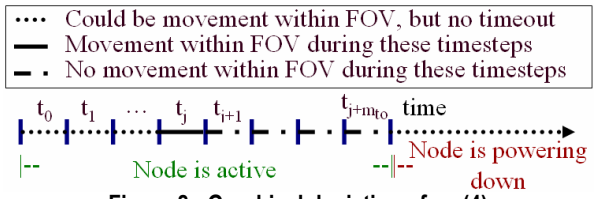


Figure 8 - Graphical depiction of eq.(4)

To complete the calculation in eq.(3), we must determine the probability of having k nodes awake given there are n neighbors. This calculation is difficult because, in reality, all nodes interact with their neighbors and affect each other's likelihood of being awake at any moment in time. Luckily, there is a lower bound we can compute as follows:

$$Pr(k\ awake\ |\ n) = \binom{n}{k} \times Pr(those\ \ k\ \ nodes\ \ awake\ |\ n) \times$$

$$Pr(remaining\ \ \ (n-k)\ nodes\ \ asleep\ |\ n)$$

$$\geq \binom{n}{k} \times Pr(1\ awake\ |\ all\ others\ \ asleep\ )^k \times \quad (5)$$

$$Pr(1\ asleep\ |\ all\ others\ \ awake\ )^{n-k}$$

$$= \binom{n}{k} \times (1 - E[st(t_s, m_{to})])^k \times E[st(t_s, (n+1)\times m_{to})]^{n-k}$$

where t_s is the time a powered-down node will sleep and $E[st(t_s, m_{to})]$ is the expected sleep fraction for a local timeout policy. This equation says that, for each of the *"n choose k"* possible ways to pick k active neighbors, the probability of having each of those neighbors either awake or

asleep can be lower bounded if one assumes that all neighbors are either all asleep or all awake. For example, the probability of having any one of those k neighbors awake must be at least as likely as the probability of having that one neighbor awake given that all other neighbors are always asleep. This can be reasoned by considering how nodes interact. If any additional node becomes active for this policy, then all other nodes currently active will only shutdown less often than they would otherwise because the recently activated node can only prevent the coordinated timeout from occurring.

A similar argument can be applied to the remaining $n-k$ neighboring nodes which are asleep in this case. The probability of any one of these $n-k$ nodes being asleep, given that all the other neighbors are awake, is upper-bounded by the case when any of those neighbors are asleep too because this can only make it more likely that the other nodes will timeout as well.

Since we now have a lower bound for the probability of k nodes being awake, we can obtain a lower bound for the expected time until shut down in eq.(3). Further, this gives an *upper bound* for the coordinated timeout policy expected sleep fraction, $E[st(t_S, m_{tO}) | n]$, given in eq.(6) and shown graphically in Fig.9 when each node has 5 neighbors.

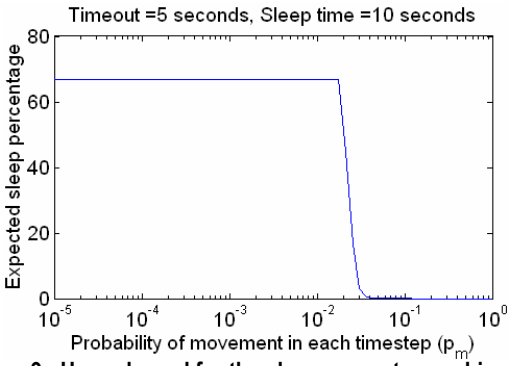


Figure 9 - Upper bound for the sleep percentage achievable with the coordinated timeout PM policy and 5 neighbors

$$E[st(t_S, m_{tO}) | n] \le min\left(\frac{t_S}{(t_S + m_{tO} \times t_{ts})}, \frac{t_S}{t_S + E[t_{sd}(m_{tO}) | n]}\right) \quad (6)$$

Next, we compare the sleep fraction achieved in simulations with that predicted analytically for the coordinated timeout policy. The sleep time used in the simulations is the same as in the local timeout case, 30 seconds, but the timeout time has been drastically reduced to 5 seconds because the neighboring nodes must also timeout for the nodes to transition to sleep mode.

The maximum sleep fraction predicted by analysis is 86%. For low camera density setups, the nodes using the coordinated timeout policy are close to achieving this maximum sleep fraction. Therefore, we can conclude that the probability of movement, p_m in eq.(4), is not exceeding the critical region for the coordinated timeout case. Of course, as camera density increases, the sleep time achieved decreases due to the increasing number of neighbors per node, so the timeout used should be decreased to cope with the increased camera density. Ideally, this timeout value should be dynamic and be decided at run-time. The analysis and implementation of a run-time adaptive distributed PM policy is left as future work.

7. Conclusion

In this paper, we have presented two new coordinated PM policies for wireless video sensor networks, discussed techniques for analytically predicting their performance, and ran extensive simulations using each PM policy to obtain power-savings estimates. Future work includes large-scale prototyping nodes using these PM techniques, as well as extending coordinated PM to other application domains.

Acknowledgments

The above work was sponsored by a fellowship for the first author from the Semiconductor Research Corporation (SRC) and by the Army Research Office (ARO) grant 9097.60.5 from Cylab at CMU.

References

[1] Fossum, E., "Active Pixel Sensors: Are CCDs Dinosaurs?" Proc. SPIE Vol. 1900, July 1993.

[2] Culler, D., et al., "Overview of Sensor Networks," IEEE Computer, Vol. 37, Aug. 2004.

[3] Bachmann, W. and Huss, S., "Efficient Algorithms for Multilevel Power Estimation of VLSI Circuits", IEEE Trans. on VLSI, Vol. 12, March 2004.

[4] Chang, J. and Tassiulas, L., "Maximum Lifetime Routing in Wireless Sensor Networks" IEEE/ACM Trans. on Networking, Vol. 12, Aug. 2004.

[5] Benini, L., et al., "A Survey of Design Techniques for System-Level Dynamic Power Management," IEEE Trans. on VLSI, Vol. 8, June 2000.

[6] Xing, G., et al., "Integrated Coverage and Connectivity Configuration for Energy Conservation in Sensor Networks," ACM Trans. on Sensor Networks, Vol. 1, Aug. 2005.

[7] Ma, H. and Liu, Y., "Correlation Based Video Processing in Video Sensor Networks," IEEE Intl. Conf on Wireless Networks, Communications and Mobile Computing, Vol. 2, June 2005.

[8] Passos, R., et al., "Dynamic Power Management in Wireless Sensor Networks: An Application-Driven Approach," Wireless On-demand Network Systems and Services, Jan. 2005.

[9] Sinha, A. and Chandrakasan, A., "Dynamic Power Management in Wireless Sensor Networks," IEEE Design and Test of Computers, Vol. 18, April 2001.

[10] Senior, A., et al., "Distributed Active Multicamera Networks," Ambient Intelligence: A Novel Paradigm, pp. 89-105, Springer Science+Business Media, Inc., 2005, New York, NY.

[11] Ellis, T., et al., "A Distributed Multicamera Surveillance System," Ambient Intelligence: A Novel Paradigm, pp. 107-138, Springer Science+Business Media, Inc., 2005, New York, NY.

[12] 802.11n ratified on January 19, 2006, http://www.enhancedwirelessconsortium.org

[13] Zorpette, G., "Super Charged [ultracapacitors]", IEEE Spectrum, Vol. 42, Jan. 2005.

[14] Anand, M., et al., "Self-Tuning Wireless Network Power Management," Wireless Networks, Vol. 11, Is. 4, pp. 451-469, Springer Netherlands, July, 2005.

[15] Gupta, R. et al., "Formal methods for Dynamic Power Management," IEEE/ACM Intl. Conf. on Computer Aided Design, Nov. 2003.

[16] Dubois, Y. and Farrell, J., "ASIC Design Considerations for Power Management in Laptop Computers," Euro ASIC, May 1991.

[17] Birman, K., "The Process Group Approach to Reliable Distributed Computing," Comm. of the ACM, Vol. 36, Dec. 1993.

[18] Rodrigues, L. and Guo, K., "Partitionable Light-Weight Groups," IEEE Intl. Conf. on Distributed Computing Systems, April 2000.

[19] Feller, W., Introduction to Probability Theory and Applications, Wiley, Vol. I, pp. 303-326.