ACM Transactions on Sensor Networks (TOSN),
Speical Issue on "New Advancements in Distributed Smart Camera Networks" (to appear in April 2014)

A Predictive Duty Cycle Adaptation Framework using Augmented Sensing for Wireless Camera Networks

PAUL J. SHIN, JOHNNY PARK, and AVINASH C. KAK, Purdue University

Energy efficiency dominates practically every aspect of the design of a wireless sensor network and duty cycling is an important tool for achieving high energy efficiencies. Duty cycling for wireless camera networks meant for tracking objects is made complex by the nodes having to anticipate the arrival of the objects in their field-of-view. The consequences of an object arriving in the view-region of a camera when it is sleeping need no elaboration. Our work presents a predictive framework to provide nodes with an ability to anticipate the arrival of objects in the field-of-view of their cameras. Our predictive framework differs from others in that the nodes whose duty cycles are increased are at least one step removed from the immediate neighborhood of the nodes where the objects are currently visible. By eliminating the need for the currently busiest nodes to also be in charge of informing their non-busy immediate neighbors to get ready for object arrival, we end up with a more robust strategy for updating the duty cycle at the nodes where the objects are highly likely to appear soon. The proposed scheme works by using an existing MAC header bit that is already in the 802.15.4 protocol and, in that sense, our anticipatory approach for notifying the nodes about the current state of the object location entails no additional expenditure of energy. Our contribution includes evaluations based on large-scale simulations as well as real experiments with an Imote2-based wireless camera network.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols–-*Protocol architecture*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, wireless camera networks, cross-layer optimization, predictive network adaptation

ACM Reference Format:
Shin, P. J., Park, J., and Kak, A. C. 2012. A Predictive Duty Cycle Adaptation Framework using Augmented Sensing for Wireless Camera Networks. *ACM Trans. Sensor Netw.* V, N, Article A (January YYYY), 32 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Central to the design of a wireless sensor network (WSN) is the need for high energy efficiency. Considering that radio broadcasting tends to be the most energy-hungry step in the operation of a WSN, one seeks to achieve high energy efficiencies by duty cycling the WSN nodes. Duty cycling is based on the straightforward rationale that if the environmental parameters that a node is monitoring stay constant over long periods of time, the node can conserve its energy resources by staying asleep much of the time.

This work was supported by Olympus Corporation under contract 200944
Author's addresses: P. J. Shin (corresponding author), J. Park, and A. C. Kak, School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Avenue,West Lafayette, IN 47907; email: {paulshin, jpark, kak}@purdue.edu

Whereas duty cycling is relatively simple to implement for WSNs meant for monitoring environmental parameters, such as air or structure quality parameters, the opposite is the case when the parameter to be monitored is of a transient nature — as is the case with wireless camera networks meant for tracking people and objects. Should a node be asleep when an object shows up in the field-of-view of its camera, at the least you would lose continuity in tracking the object. Obviously, then, the latency introduced by duty cycling in recording the parameters of interest must be minimized when such parameters are allowed to be transient.

The interest in using duty cycling for enhancing the energy efficiency of WSNs has led to several contributions on the incorporation of the same in the MAC protocol layer [Rhee et al. 2008; Buettner et al. 2006; Polastre et al. 2004; El-Hoiydi and Decotignie 2004; Ye et al. 2002]. But, as we mentioned above, any static approach to duty cycling would be found wanting when the parameters that need to be monitored by a WSN are transient. So the past several years have also witnessed contributions that incorporate dynamic duty cycling in the MAC layer that can adapt to the variations in the temporal properties of the parameters of interest. These contributions have resulted in the MAC protocols such as TMAC [Van Dam and Langendoen 2003], AMAC [Nam et al. 2006], DSMAC [Lin et al. 2004], and CMAC [Liu et al. 2009].

That brings us to an examination of the currently available dynamic approach to duty cycling from the standpoint of what is needed for wireless camera networks. In the research contributions we have cited above, the dynamic duty cycle adaptation at a node is based on the detection of a change in the *current* traffic conditions at the node. We claim that such passive duty cycle adaptation schemes are not appropriate for event-driven WSNs, such as wireless camera networks (WCNs) intended for tracking humans and objects in motion. Our claim is based on the observation that the delay between the time an event occurs and the time a new duty cycle regime becomes effective may be unacceptable if the event corresponds to a fast moving object being tracked by a camera network.

What we really need for wireless camera networks are duty cycle adaptation strategies that can anticipate the arrival of events of interest *provided the methods used for anticipation entail only minimal communication overhead.* The condition stated in italics is important since a trivial anticipation strategy consisting of the currently active nodes merely broadcasting their activity status to all the neighboring nodes is not likely to work effectively in practice. The currently active nodes are likely to be the members of clusters that are engaged in observing the target and calculating its motion parameters. They cannot be expected to also be responsible for communicating with the non-cluster members. Even if such a simpleminded approach to duty cycle adaptation was made to work under certain experimental conditions (such as when we have a small number of slowly moving objects), it would not scale up properly as the event activity levels increase. Note that the intra-cluster traffic tends to be intense and bursty. That increases the odds that random communications from cluster members to non-cluster members in order to lend to the latter the ability to anticipate future traffic are likely to fall prey to the expected high levels of contention. The packet collision/loss rate in wireless networks, in general, increases *exponentially* due to contention as the traffic increases. Therefore, introducing additional traffic for explicit notification of the current object state to non-cluster nodes also has the potential to significantly increase the loss of critical information related to the tracking task. This may result in severe degradation of the tracking performance, reduced tracking accuracy, and possibly frequent tracking failures.

These observations have motivated us to develop a duty cycle adaptation framework that requires only minimal communication overhead. Our adaptation framework enables a node to infer the current state of an object of interest that is *beyond* its sensing

ACM Transactions on Sensor Networks (TOSN),
Speical Issue on "New Advancements in Distributed Smart Camera Networks" (to appear in April 2014)

capability. Our proposed approach, called PDCA (for *Predictive Duty Cycle Adaptation*) is a predictive network adaptation framework where a node can adapt its parameters *in advance* of a moving target actually showing up in its field-of-view and do so with no additional communication cost. The engine that drives the adaptation is an event/object tracker that can interact with all of the layers of a protocol stack. The interaction between any of the protocol stack layers and the tracker creates inputs for the tracker that can be used to update the state of the object being tracked. The updated state thus calculated can subsequently be retrieved by all the layers of the protocol stack. Our approach to modeling the interaction between the tracker module and the layers of the protocol stack allows for online cross-layer optimization to be carried out while an object is being tracked.

Our framework for duty cycle adaption in anticipation of upcoming high traffic is made possible by the notion of *augmented sensing* that consists of *direct sensing* and *indirect sensing*. Whereas direct sensing refers to sensing by a node's own camera, indirect sensing is defined as obtaining a measurement (or any object state information inferred therefrom) from the sensing capability of some other nodes via a communication channel. We show how indirect sensing can be achieved by using a single bit embedded in the MAC header of all outgoing packets. We refer to this one bit as the *Explicit Event Notification* (EEN) flag. Since this flag can be set within the Frame Control Field (FCF) of the MAC header, which is available in most standard MAC protocols such as 802.15.4, our proposed method does not require any modification of the structure of the packet formats of the existing protocols. Moreover, since this flag is set in the packets that are supposed to be transmitted anyway, our proposed method does not involve any additional communication overhead.

The event tracker at each node uses a Kalman filter to aggregate all the measurements obtained through augmented sensing in order to keep track of the object of interest. Thus, all nodes — even those that are not currently seeing the object — can estimate the current state of the object and make predictions of the future state of the object. This allows each node to adapt its duty cycling regimen in anticipation of the arrival of the object in the near future.

A noteworthy aspect of the duty cycle adaptation framework presented in this paper is that it allows different nodes to operate with different duty cycles. For obvious reasons, this creates challenges in any communication between the nodes, especially for those modes of communication that do not call for handshaking with ACK. The work we present here includes a novel approach for nodes with different duty cycles to engage in communications.

Our proposed framework for duty cycle adaptation, however, does require that the camera nodes that are currently seeing the object generate radio traffic. This assumption is easily satisfied by the wireless camera networks designed for tracking objects. The nodes of such networks must engage in collaborative computing within clusters for the calculation of object state, which naturally results in the sort of radio traffic our framework depends on.

PDCA can be used as a plug-in/add-on to any synchronous MAC protocol for WSN. The work we present here shows how the performance of a synchronous MAC protocol can be improved when PDCA is also deployed for duty cycle adaptation. The performance numbers we present in support of this conclusion are based on energy efficiency, tracking performance, and Time-Bounded Parameter Estimation Accuracy (TIBPEA) [Shin et al. 2007]. These performance comparisons will be based on both large-scale simulations and, even more importantly, on real experiments with an Imote2-based wireless camera network.

In the rest of the paper, we first review in Section 2 some unique features of wireless camera networks. Section 3 then provides a review of the relevant literature on adaptive duty cycling in WCNs. The new PDCA framework is presented in Section 4 and its application to duty cycle adaptation in Section 5. The performance evaluation of the framework is presented in Section 6. As already mentioned, our performance evaluation is based on large-scale simulations and on a real Imote2-based testbed. Section 7 then concludes the paper.

2. WIRELESS CAMERA NETWORKS

As a sub-category of the more general WSNs, the WCNs tend to be event-driven, in the sense that their operation is frequently triggered by the detection of an event of interest by one or more camera nodes. In addition to their being event-driven, WCNs differ from the typical WSNs also from the fact the cameras are directional sensors. In this section, we focus on two unique characteristics of WCNs that are relevant to our work. For a more comprehensive survey on camera sensor networks, the reader is referred to [Soro and Heinzelman 2009] and [Charfi et al. 2009].

2.1. Traffic Patterns for Event-Driven Collaborative Processing

WCNs differ from the more traditional WSNs in that intense and bursty radio traffic is generated in the vicinity of the events as the camera nodes participate in a collaborative interpretation of the event. As a result, the network-wide variation of the traffic pattern in WCNs is much larger than that in the conventional WSNs where the traffic generation tends to be periodic and the pattern remains unchanged in the course of time.

With regard to the burstiness of the local traffic caused by collaborative interpretation of the sensed data, it is obviously made necessary by the limited computational power and sensing capability of the nodes. The nodes that participate in such collaborative computations are said to form a cluster. A typical cluster based approach to collaborative sensing [Medeiros et al. 2007] includes operations such as 1) cluster formation; 2) cluster leader election; 3) cluster propagation, with cluster leader re-election whenever necessary; 4) estimation of the properties of the objects of interest collaboratively; etc. All of these phases of cluster based computing require highly bursty communications.

Such collaborative nature of processing visual data among the nodes in a WCN is in contrast to the nodes in a traditional WSN where scalar measurements are acquired by each node independently and these measurements are simply aggregated in the network in order to remove redundancies in data transmission. An example of the traditional WSNs would be a wireless network meant for monitoring the environment for, say, the air quality.

2.2. Quality of Service (QoS)

The QoS criteria for WCNs must of necessity be more stringent as compared to those for the more conventional WSNs. Any latencies introduced by, say, duty cycling and any information loss caused by high traffic contention — traffic contention is more of an issue for WCNs given the bursty nature of the traffic — is likely to result in more serious performance degradation. Excessive latencies may, for example, cause to loose track of a target object.

Achieving a high QoS by simply over-provisioning the nodes for performance requirements so that they can cater to the expected peak traffic is obviously not feasible since the wireless camera nodes are generally resource constrained. It is also not feasible in such networks to simply reduce the duty cycle at the nodes for energy requirements since that would adversely affect the QoS. Rather, in order not to lose critical information related to the event and also in order to process (in some cases, collaboratively) the sensed data in real-time, these nodes must be able to quickly alter their network parameters (e.g., the duty cycle of radio or the sensing rate). These requirements pose unique challenges for designing communication protocols that are suitable for WCNs.

3. LITERATURE REVIEW

The need for network adaptation arises from the fact that the different nodes in a network may experience different degrees of traffic and the fact that each node must conserve its energy budget to the maximum extent possible. Even when a WSN is used for just environmental monitoring, typically each node reports its measurement to the base station through a tree-like routing path. Consequently, the amount of traffic each node experiences depends on its position in the routing path. The closer a node is to the base station, the more traffic it will see. Such nonuniformities become all the more severe in WCNs since an event of interest is observed and processed by only a small subset of the nodes at any given time.

In the context of using a WSN to track animals with passive infrared motion sensing, the Frisbee model [Cerpa et al. 2001; Farina et al. 2005] was proposed to dynamically set a circular zone of a pre-defined radius around the *current* location of the target being tracked by the network. The nodes within the zone are kept in fully active state (meaning at 100% duty cycle) while those outside the zone are made inactive (meaning at 0% duty cycle) in order to conserve their energy. This circular zone moves with the target. The system uses a wakeup signal on a different low-power radio as a mechanism to wake up the nodes that are currently asleep. Therefore, as the target moves, a so-called “wake-up wavefront” propagates in the direction of the movement of the target. This approach, however, requires that the nodes be equipped with special hardware for generating wakeup signals and for responding to the same. Also, using a wake-up radio to activate a transceiver forces *all* nodes within one-hop range to turn on their transceivers regardless of what the radius of the Frisbee zone is set to. The Frisbee model also requires a small subset of nodes to always be on in order to initiate the process of object tracking.

Instead of using a special wake-up radio, the Probing Environment and Collaborating Adaptive Sleeping (PECAS) method presented in [Gui and Mohapatra 2004] requires nodes that detect an event of interest to proactively broadcast wakeup packets. The nodes that receive such wakeup packets (these would be within one-hop radius of the event detecting nodes) broadcast a Prepare message to their neighbors. This allows for the nodes within a two-hop radius of the event-detecting nodes to change their duty cycle to full on. This type of a proactive wakeup mechanism, however, results in a large number of wakeup packets generated in a local area, which causes increased packet collisions and information loss. Both this scheme and the Frisbee approach entail spatial areas of fixed size in relation to where the events are taking place for the change in the activity levels at the nodes.

Instead of waking up all the nodes within a fixed range, the approach described in [Tseng and Lu 2009] *predicts* a smaller set of nodes that may next detect the object through their motion sensors and then wakes up only those nodes. To make these predictions, object movement logs over time are stored in a database and the object motion patterns predicted on the basis of those logs. Predictions obtained in such a manner may not work should the object exhibit erratic motions as a function of time. We should also mention other node activation schemes, such as those in [Zhang and Cao 2004; Xu et al. 2001], for structured networks that adaptively activate/deactivate nodes. However, these suffer from shortcomings similar to those described above.

In addition to the previous approaches, we should also mention the contribution in [Mir et al. 2010] that presents the EC-MAC (Event-Centric MAC) protocol for selective node activation. This MAC layer protocol — which operates at a level in the protocol stack below those of the previous approaches — is a hybrid scheme that uses asyn-

ACM Transactions on Sensor Networks (TOSN),
Speical Issue on "New Advancements in Distributed Smart Camera Networks" (to appear in April 2014)

chronous random access and TDMA to facilitate efficient and reliable communication for cluster-based collaborative data processing and the forwarding of packets from the cluster members to the cluster head within each active region. EC-MAC defines an active region as a set of nodes that are *already* detecting an event, and it is based on a tree organization of the cluster members vis-a-vis the cluster head. The tree relationship between the nodes in an active region is brought about through the execution of a tree construction algorithm that can degrade the performance of the system by increasing the network latency. In addition to EC-MAC, there are other MAC layer protocols that dynamically adapt the duty cycle of the radio based on synchronous [Van Dam and Langendoen 2003; Lin et al. 2004; Nam et al. 2006] and asynchronous [Hurni and Braun 2010; Anwander et al. 2010; Jurdak et al. 2007] schedules.

It bears repetition to mention that all the duty cycle adaptation schemes mentioned above (including also other adaptive resource allocation methods such as the one described in [Mainland et al. 2005]) adapt their parameters on the basis of the *current* network conditions related to the current event location, the current buffer state, the incoming traffic rate, and so on. Such reactive adaptation methods entail an inherent delay between the time of detection and the time of adaptation. By contrast, our proposed approach allows a node to predict the future state of an event and adapt its network parameters *before* the arrival of the event. Moreover, unlike existing proactive schemes that require wake-up messages to be broadcast explicitly, our proposed method does not incur any communication overhead as it is carried out with implicit notification embedded in the MAC header of all outgoing packets.

4. OBJECT TRACKING VIA AUGMENTED SENSING

In this section, we present a novel framework for object tracking within the network protocol stack of WCNs. The proposed object tracker resides in a separate module in each node that can interact with all of the layers of a protocol stack. The interaction between any of the protocol stack layers and the tracker creates inputs for the tracker that can be used to update the state of the object being tracked. The updated state thus calculated can subsequently be retrieved by all the layers of the protocol stack. Such interaction between the tracker and the layers of the protocol stack allows for online cross-layer optimization to be carried out simultaneously while an object is being tracked.

Consider a WCN shown in Figure 1. The communication range and the field-of-view of each camera node are shown, respectively, in Figures 1(a) and (b). The goal of this camera network is to track an object of interest depicted as the star in Figure 1(c). As the object moves, the cameras that can detect this object form a cluster to track the object collaboratively. The data aggregated within the cluster is then delivered to the base station (or sink) through multi-hop communications as shown in Figure 1(c). The nodes labeled E and F represent those cameras that are currently able to see the object and are actively participating in the data aggregation. Thus, the duty cycle of these nodes must be set sufficiently high in order to carry out the collaborative object tracking. Since it is likely that the nodes A and B will soon see the object based on the expected future location of the object, their duty cycle should be increased in order to achieve a low-latency condition *prior to* the object becoming visible to them. The methods with which we use to achieve this will be discussed in detail in Sections 4.1 to 5.5. The nodes B, H, C, and D are those that are actively participating in delivering the aggregated data to the base station. The duty cycle at these nodes would also need to be sufficiently high so that the packets containing the information of the object can be delivered to the base station as quickly and as reliably as possible. Our proposed method to achieve this will be the topic of Section 5.6.

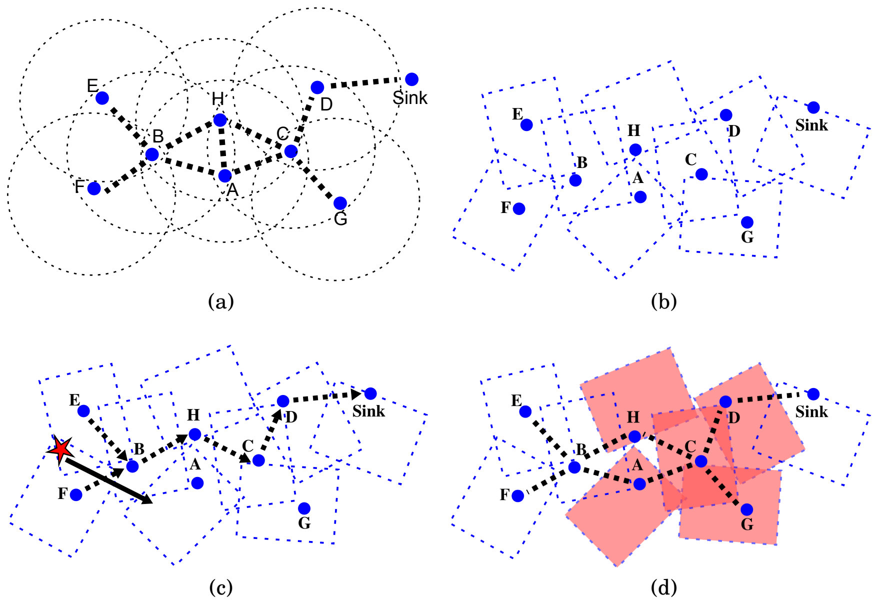


Fig. 1: A depiction of a WCN engaged in tracking a moving object. The dotted circles and lines in (a) represent the communication range and connectivity among nodes, respectively, and the dotted rectangles in (b)-(c) represent the sensing field of nodes. The red star in (c) indicates the object of interest and the black solid arrow its moving direction. The black dotted arrows in (c) indicate the routing path of the packets triggered by the detection of the object. The augmented sensing field of node C is shown in (d) as an example, which is the union of the sensing field of node C and its communication neighbors illustrated as red regions.

To enable predictive network adaptation, the nodes where the objects are highly likely to appear soon must be notified so that they can get ready to handle the imminent increase in radio traffic. If such notification is carried out through explicit transmission of packets, it will entail not only additional energy consumption but more importantly increased traffic in the nodes that are engaged in object tracking. Our proposed approach therefore does not involve transmitting explicit notifications by the currently busiest nodes. Instead our approach employs an implicit notification embedded in the MAC header of all outgoing packets so that the receiving nodes can infer the state of the object and adapt the network parameters accordingly. Since the notification is set in the packets that are supposed to be transmitted anyway, our method incurs no additional communication overhead. In Section 4.1, we will describe in detail how the implicit notification is carried out by the notion of *augmented sensing*.

The aforementioned online network optimization technique requires a prediction of the state of the object and a proper metric to estimate the probability of the object appearing at a node at a given time. In Section 4.3, we will present how to keep track of the current position of the object and make a prediction on the state of the object.

4.1. Augmented Sensing

Sensing generally refers to an action of obtaining measurements from a sensor. In the context of WCN, sensing can be interpreted as obtaining object measurements from the images acquired by a camera attached to a wireless mote. We will refer this type of sensing as *direct sensing*. *Indirect sensing,* on the other hand, refers to obtaining measurements (or any object state information inferred therefrom) from the sensing capability of the neighboring nodes via a communication channel. We will use the term *augmented sensing* to include both direct sensing and indirect sensing.

Indirect sensing is usually in a summarized or compressed form since communication is expensive in WSNs. The summarized measurements of an object (e.g., center of mass and direction) can be made to known to other nodes by explicitly transmitting packets that contain the object information. The receiving nodes can extract the current state of the object and adapt their parameters, if necessary, in anticipation of the arrival of the object. As we mentioned earlier, this approach is not likely to work in practice because it creates additional communication overhead for the already busiest nodes in the network that are currently engaged in object tracking. These nodes cannot be expected to also be responsible for broadcasting the object state information to other neighboring nodes in the network. Therefore our proposed framework employs indirect sensing that does not require any communication overhead. The indirect sensing in our framework is a single-bit flag embedded in the MAC header of all outgoing packets. We refer to this one bit as *Explicit Event Notification* (EEN) flag[1]. Since a node currently engaged in object tracking is bound to generate a large number of packets, embedding a binary flag in all outgoing packets will effectively notify the neighboring nodes about the presence of the object inside the sensing field of the node without incurring any additional communication traffic. A node receiving/overhearing a packet with the EEN bit set to 1 can assume that the object of interest is located somewhere in the field-of-view of the node that sent the packet. The augmented sensing field of a node therefore is the union of its own field-of-view and the field-of-view of its one-hop neighbors. Figure 1(d) illustrates the augmented sensing field-of-view of node C. As we will describe later, a Kalman filter at each node aggregates all the measurements obtained through augmented sensing in order to keep track of the object.

In order to carry out indirect sensing, a node must know about the field-of-view of its one-hop neighbors. For that purpose, we assume that the cameras in the network have been localized and calibrated and that the field-of-view of each camera is available in the form of a 3-tuple (i, z, R) where i identifies the node (i_{self} is the local node address), z corresponds to the center of its field-of-view, and R is an ellipsoid that approximates the area of the field-of-view. During an initialization stage, each node broadcasts its field-of-view information to its one-hop neighbors. When a node receives the field-of-view information from its neighbors, it simply stores them in a list within the tracker module.

Since EEN can be set within the FCF field of the MAC header, which is available in most standard MAC protocols such as 802.15.4, our proposed method does not require any modification of the structure of the packet formats of existing protocols. Moreover, since EEN is included in the packets that are supposed to be transmitted anyway, our proposed method does not incur any additional communication overhead. Obviously one could allocate additional bits to embed more descriptive information about the object. In fact, the FCF field of the MAC header of the current implementation of 802.15.4 MAC protocol in TinyOS takes only 7 bits out of available 16 bits, leaving us with up to 9 bits that could be used for EEN. In this paper, we use only one bit for EEN to demonstrate that there is a significant performance improvement by our proposed PDCA framework even with the coarsest indirect sensing achieved by a single bit.

For obvious reasons, indirect sensing would be most effective if each node checks the EEN bit for all receiving packets, including those packets not intended for the node.

[1]EEN can be thought of as a generalized and implicit form of ECN (Explicit Congestion Notification) used in TCP/IP.

This can be easily implemented by checking the EEN bit first at a protocol stack *before* the destination address check in the MAC layer. Thus, it does not entail unnecessary energy consumption that usually happens in typical packet overhearing. Since we only intend to detect whether or not the EEN bit is set, we refer to this process as *event packet detection*.

4.2. Tracker Design

It is reasonable to assume that each node can acquire multiple observations of the object via direct and/or indirect sensing. Each measurement may be represented by the expected location of the object along with its uncertainty in the form of a covariance matrix. The goal of the discussion that follows is to show how all the measurements acquired sequentially as the object is being tracked can be used in a recursive framework to predict as to what nodes are likely to see the object next with what probability. There is a number of recursive estimation methods available including various types of Particle filters [Doucet et al. 2000; Coates 2004], but we chose to use Kalman filter [Welch and Bishop 1995] because of its low computational and memory requirement.

Note that the Kalman filter employed in this paper is one of the simplest forms with a linear dynamics and measurement model. We fully realize that such a simple model is not adequate to capture various object movements that may occur in different applications. Our intention is to show that even with such a simple tracker, a significant performance gain can be made possible using our proposed PDCA framework. Obviously the type of tracker used in the system would impact the level of performance gain, and such evaluation is beyond the scope of this paper.

4.2.1. System Model and Kalman Filter Equations. Each node that is currently engaged in observing and tracking the object of interest will create a state vector for the object. When a new object is detected within the augmented sensing field-of-view of a node, the state vector of the object is initialized with the initial object observation. Subsequently, the node uses the Kalman filter equations to update the state vector. This updated state vector is then used to make a prediction about where the object will likely to appear next.

We model the object state as a 4-D vector that consists of the object position (x_k, y_k) at a discrete time instant k and its velocity $(\dot{x}_k, \dot{y}_k)$. That is, the state vector is given by

$$\mathbf{x}_k = [\, x_k \;\; y_k \;\; \dot{x}_k \;\; \dot{y}_k \,]^T .$$

The system dynamics are modeled by

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k + \delta_k \dot{x}_k + \frac{a_x}{2}\delta_k^2 \\ y_k + \delta_k \dot{y}_k + \frac{a_y}{2}\delta_k^2 \\ \dot{x}_k + a_x \delta_k \\ \dot{y}_k + a_y \delta_k \end{bmatrix},$$

where δ_k is the time elapsed between two successive observations. That is, if the k^{th} observation was acquired at time t_k, the observation $k+1$ is acquired at time $t_{k+1} = t_k + \delta_k$. The event acceleration (a_x, a_y) is modeled as white Gaussian noise with covariance matrix Q_k. Then, the system dynamics can be represented as

$$\mathbf{x}_{k+1} = F_k \mathbf{x}_k + W_k \mathbf{w}_k$$

where

$$\begin{aligned} F_k &= \begin{bmatrix} I_{(2\times2)} & \delta_k I_{(2\times2)} \\ 0_{(2\times2)} & I_{(2\times2)} \end{bmatrix}, \\ W_k &= \begin{bmatrix} \frac{\delta_k^2}{2} I_{(2\times2)} & \delta_k I_{(2\times2)} \end{bmatrix}^T, \end{aligned}$$

ALGORITHM 1: Object tracking via indirect sensing using KF in the tracker module: Cross-layer interaction from MAC and application layer

```
**< Application layer >**
$\mathbf{z}_{direct} \triangleq$ a measurement from camera in a form of coordinates
$R_{direct} \triangleq$ the covariance matrix associated with the measurement $\mathbf{z}_{direct}$
repeat
    if a new sensing result is available then
        if an object is detected within the sensing field at time $t_k$ then
            Provide tracker module the new measurement $(\mathbf{z}_{direct}, R_{direct})$;
        else                                                    ▷ No object is detected
            Provide tracker module a $NULL$ measurement;
until Node is running;

< MAC layer >
$packet(i) \triangleq$ a packet transmitted by node $i$
$NodeId[Size] \triangleq$ an array to store the list of node IDs
$isDetecting \triangleq$ a boolean indicating whether the node is currently detecting an object or not
$T_{timeout} \triangleq$ the periodic time interval to send $NodeIDs$ to tracker module
A timer is set to expire at every $T_{timeout}$;
repeat
    if a new $packet(i)$ is available with EEN bit set then     ▷ Indirect sensing
        Insert $i$ into storage $NodeId$;
    if timer is expired $||$ $NodeId$ is full then
        Send storage $NodeId$ to tracker module and Reset timer and storage $NodeId$;
    if a packet is ready to be sent $\&\&$ channel is clear then
        EEN bit of the packet $\leftarrow isDetecting$;
until Node is running;
```

and $\mathbf{w}_k = [\, a_x \;\; a_y \,]^T$ is the process noise vector with covariance matrix Q_k.

Each direct sensing measurement consists of the approximate location of the object along with with its uncertainty as given by the covariance matrix associated with the state vector. The measurement model can be described by

$$\mathbf{z}_{k+1} = H_{k+1}\mathbf{x}_{k+1} + \mathbf{v}_{k+1},$$

where $H_{k+1} = [\, I_{(2\times2)} \;\; 0_{(2\times2)} \,]$ and $\mathbf{v}_{k+1}$ is the measurement noise, assumed white Gaussian with covariance matrix R_{k+1}.

Let $\hat{\mathbf{x}}_{k+1|k}$ and $\hat{\mathbf{x}}_{k|k}$ be the predicted and the previously estimated state vectors, and similarly, $P_{k+1|k}$ and $P_{k|k}$ the predicted and the previously estimated covariance matrices. Then, the time update equations of the Kalman filter are given by

$$\hat{\mathbf{x}}_{k+1|k} = F_k\hat{\mathbf{x}}_{k|k} \quad (1)$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + W_k Q_k W_k^T. \quad (2)$$

4.2.2. State Updates by Indirect Sensing . Exactly the same Kalman Filter that was described in the previous subsection is used to estimate the state vector using indirect sensing measurements. Note that when a node detects a packet with its EEN bit set to 1, then the MAC layer can provide the ID of the sender to the tracker module. The tracker first converts the ID of the sender into $(\mathbf{z}, R)$ using the field-of-view information obtained during the initialization stage. The Kalman filter then uses $(\mathbf{z}, R)$ as an input to update the state vector of the object. Figure 2 illustrates this process.

ALGORITHM 2: Object tracking via indirect sensing using KF in the tracker module: Cross-layer interaction at the tracker module

```
< Tracker Module >
$\mathbf{z}_{indirect}(i) \triangleq$ the center of the sensing field of node $i$
$R_{indirect}(i) \triangleq$ the ellipsoidal approximation of the sensing field of Node $i$ represented as a covariance matrix
repeat
    if an array $NodeId[N]$ is received from MAC layer then                          ▷ Indirect sensing
        for $n := 0$ to $N - 1$ step 1 do
            $Kalman(\mathbf{z}_{indirect}(NodeId[n]), R_{indirect}(NodeId[n]))$        ▷ calls Function
        else if a new $(\mathbf{z}_{direct}, R_{direct})$ is received from application layer then   ▷ Direct sensing
            if $\mathbf{z}_{direct} \neq NULL$ then
                Set $isDetecting \leftarrow True$ in the MAC layer;
                $Kalman(\mathbf{z}_{direct}, R_{direct})$                                ▷ calls Function
            else
                Set $isDetecting \leftarrow False$ in the MAC layer;
until Node is running;
```

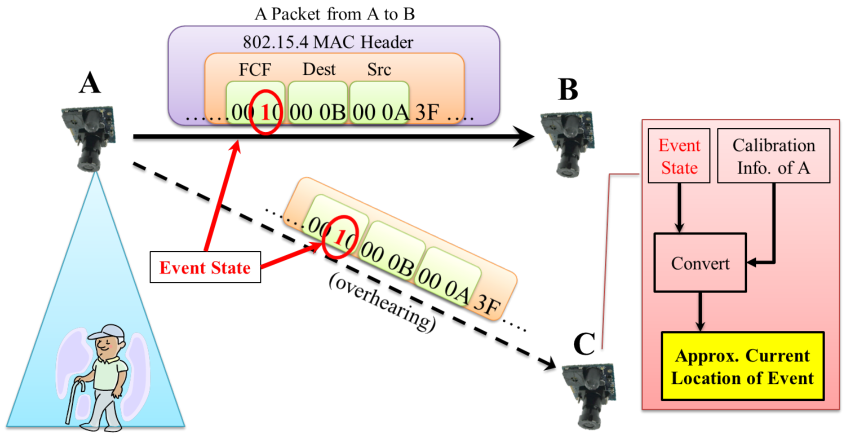


Fig. 2: An illustration of how the event packet detection is used not only to get advance notice of an approaching target but also to decide whether the target is close enough for any changes to the local communication parameters. Node A is currently detecting an object and wants to collaborate with node B. The outgoing packets from node A have their EEN bit set. Node C, which is within a single hop range of node A, detects one such packet. Then, node C uses this information to form an estimate of the current position of the object.

The estimation of the state vector of an object by indirect sensing only (i.e., by event packet detection) poses an interesting challenge: How should the node that detects the EEN bit associate observation uncertainty with the object position at the sender of the EEN bit? The approach taken in our work is to assume that the mean position of the object is at the center of the field-of-view of the sender. We further assume that the covariance of the object position can be approximated by the area of the field-of-view of the sender. In other words, an indirect sensing measurement obtained by a packet sent by node i takes a form of a Gaussian distribution $(\mu(i), \Sigma(i))$, where $\mu(i)$ is the center of the field-of-view of node i and $\Sigma(i)$ the covariance matrix approximated by the area of the field-of-view of node i. This simple Gaussian approximation enables our predictive duty cycle adaptation scheme to be applied even to the extremely resource-constrained wireless embedded devices. One could use a nonlinear estimator such as a Particle filter at the cost of larger resource overhead.

We want to note that a naive approach of updating the object state vector *each time* a node receives an indirect sensing measurement may result in inaccurate and biased state estimation. As mentioned before, as an object is being tracked, all the nodes in the vicinity of the object can acquire measurements via direct and indirect sensing. If a cluster is created among the nodes that are detecting the same object, one of them will be elected as the cluster head and the rest its members. Depending on the role of a node in a cluster, the traffic rate generated by each node could vary significantly; mostly the cluster head generates more traffic for reporting the updated object states to the base station and for performing various cluster operations. Therefore, if a node updates its tracker *each time* it receives an indirect sensing measurement (i.e., detects an EEN-set packet from one of the clustering nodes), then the estimation of the tracker would likely be biased toward the center of the sensing field of the cluster head.

To resolve this issue, we devise in the MAC layer an array for storing the node IDs and a timer with a timeout threshold $T_{timeout}$. The MAC layer stores the ID of the sender of an EEN-set packet in the array unless the ID is already in the list. Whenever the timer expires or the array is full, the MAC layer sends the list of node IDs to the tracker module and resets the list. The tracker then converts the node IDs into a set of measurements $(\mathbf{z}, R)$ and updates its states.

Of course, this time-window based approach introduces a latency in the Kalman filter update. However, we argue that the benefits of avoiding estimation bias and instability far outweigh the slightly increased latency in the Kalman filter update.

One strategy for reducing the effects of the increased latency is to set the EEN bit of a packet only *after* making sure the channel is clear right before the packet is transmitted. The nodes receiving this packet can estimate the time that the measurement was actually taken by subtracting the expected transmission delay of the packet. Since the transmission delay is consistent for the packets with the same length, the detector can estimate the measurement time quite accurately.

If a packet with its EEN bit set is detected along with its corresponding time stamp, and a time synchronization is maintained among the nodes, we can easily obtain a reasonably precise time measurement and accurately compute the time increment δ_k [Kusy et al. 2006]. Even when the measurement time is not accurate, note that the received measurement in the MAC layer via indirect sensing is already a good approximation to the actual position of the object. Thus it is reasonable to assume that any inaccuracies caused by time jitter in indirect sensing can be expected to be negligible.

Algorithm 1 and 2 summarize the Kalman filter based tracking at each node.

4.3. Spatio-Temporal Event Probabilities

Given an event (or object) j at a time instant t_k, the corresponding spatio-temporal event probability (STEP) distribution at a particular position $\mathbf{u}$, denoted as $S^j_{k+1|k}(\mathbf{u})$, is defined as *the probability of the predicted position of the event j at position $\mathbf{u}$ at time $t_{k+1} = t_k + \delta_k$* , where δ_k is the prediction interval. That is, let $\mathbf{p}^j_{k+1|k}$ be the predicted position of the event, then the STEP at position $\mathbf{u}$ at t_{k+1} is given by

$$S^j_{k+1|k}(\mathbf{u}) = Pr(\mathbf{p}^j_{k+1|k} = \mathbf{u}). \tag{3}$$

Note that $S^{j}_{k+1|k}(\mathbf{u})$ and $\mathbf{p}^{j}_{k+1|k}$ correspond to the prediction at a future time instant t_{k+1}. The prediction interval δ_k should be determined to be larger than the time needed for a node to change its network parameters such as the duty cycle and the time needed for the change to actually take place so that the node can complete the adaptation process before the event actually occurs at the node.

Once the tracker module receives a list of node IDs from the MAC layer and completes all updates, the tracker estimates the STEP distribution by using its prediction module to make a prediction of the future position of the target. Assuming the current estimated state is $\hat{\mathbf{x}}_{k|k}$ after completing all updates, the predicted position of the target $\mathbf{p}^{j}_{k+1|k}$ for time t_{k+1} is given by

$$\mathbf{p}^{j}_{k+1|k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \hat{\mathbf{x}}^{j}_{k+1|k}.$$

Since the prediction interval δ_k is not known at time t_k, the prediction is carried out using a pre-defined δ_k. The STEP at position $\mathbf{u}$ at t_{k+1} then can be estimated according to using Eq. (3).

4.4. Network Parameter Adaptation

Network parameter adaptation at a node may be carried out either on the basis of the probability of detecting the object in the field-of-view of the node or on the basis of the expected arrival time of the object. In this section, we will first show how each node can estimate the probability of detecting an object in its field-of-view by computing how much the STEP overlaps with the field-of-view. We define this probability as the *Future Event Detection Probability* (FEDP) at a node. We will then show that how each node can estimate the mobility-based time-of-arrival of the event, which is defined as the *Expected Time-of-arrival of Event* (EToE).

4.4.1. Future Event Detection Probability. Given the STEP distribution of an event j at time t_k, each node predicts how likely the event will occur within its field-of-view at time t_{k+1} using the prediction module of the tracker. The probability that an event j will occur within the sensing field of node i at time t_{k+1}, $S^{(j,i)}_{k+1|k}$, is computed by:

$$\begin{aligned} S^{(j,i)}_{k+1|k} &= Pr(\mathbf{p}^{j}_{k+1|k} \in G(i)) \\ &= \int_{\mathbf{u}\in G(i)} Pr(\mathbf{p}^{j}_{k+1|k} = \mathbf{u}) d\mathbf{u} \\ &= \int_{\mathbf{u}\in G(i)} S^{j}_{k+1|k}(\mathbf{u}) d\mathbf{u}, \end{aligned}$$

where $G(i)$ denotes the sensing field of the node i as illustrated in Figure 4(a). This future event detection probability (FEDP) is computed as the integration of STEP over the field-of-view of a node. This figure also illustrates that the STEP distribution associated with the object being tracked may span the field-of-view of multiple cameras.

The state transition diagram shown in Figure 3 describes how FEDP is computed with Kalman filter. While the state of the Kalman filter is updated whenever a new set of measurements is available, the network parameter adaptation block also computes a new FEDP based on the updated states in the Kalman filter. According to the new FEDP, the node determines a new parameter value for adaptation in each individual layer. The FEDP at a node is approximated by dividing the field-of-view into m

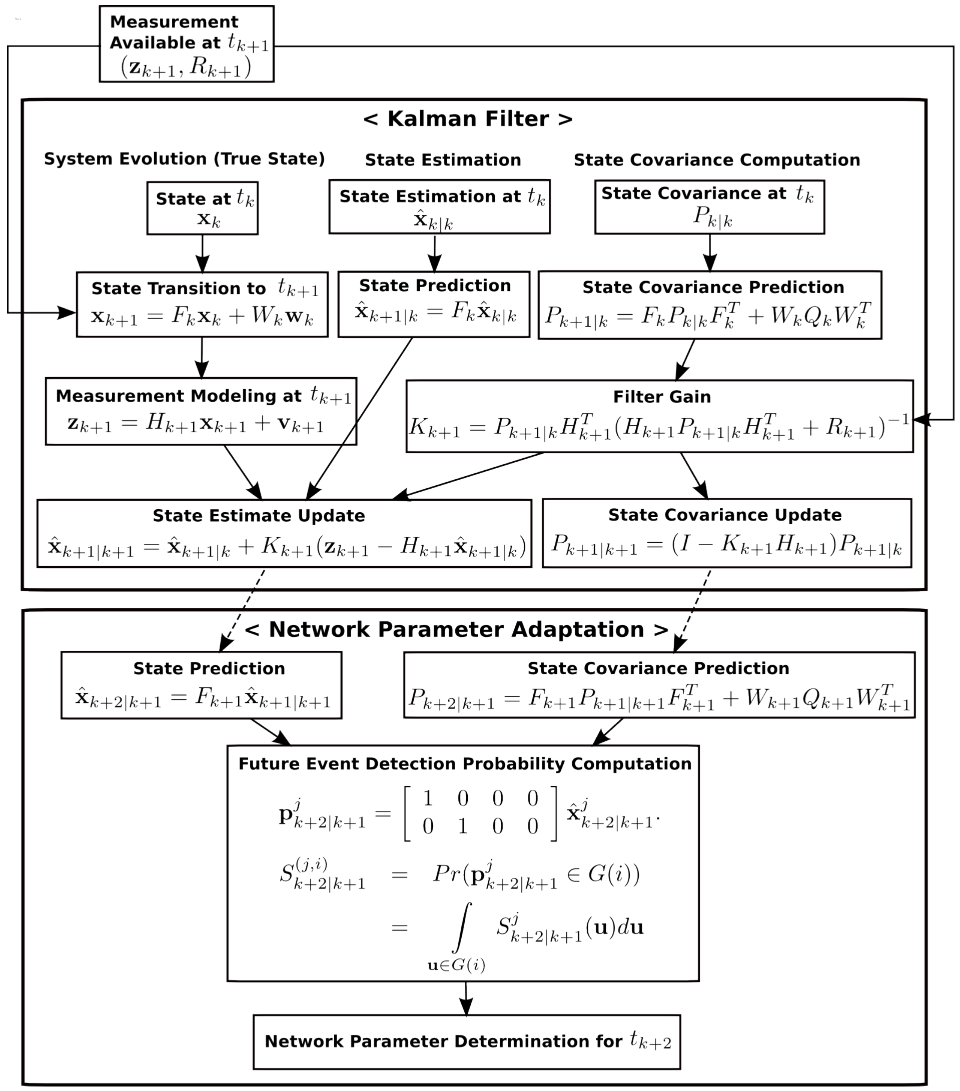

Measurement Available at t_{k+1} $(\mathbf{z}_{k+1}, R_{k+1})$
< Kalman Filter >
System Evolution (True State)
State Estimation
State Covariance Computation
State at t_k $\mathbf{x}_k$
State Estimation at t_k $\hat{\mathbf{x}}_{k|k}$
State Covariance at t_k $P_{k|k}$
State Transition to t_{k+1} $\mathbf{x}_{k+1} = F_k\mathbf{x}_k + W_k\mathbf{w}_k$
State Prediction $\hat{\mathbf{x}}_{k+1|k} = F_k\hat{\mathbf{x}}_{k|k}$
State Covariance Prediction $P_{k+1|k} = F_kP_{k|k}F_k^T + W_kQ_kW_k^T$
Measurement Modeling at t_{k+1} $\mathbf{z}_{k+1} = H_{k+1}\mathbf{x}_{k+1} + \mathbf{v}_{k+1}$
Filter Gain $K_{k+1} = P_{k+1|k}H_{k+1}^T(H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1})^{-1}$
State Estimate Update $\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + K_{k+1}(\mathbf{z}_{k+1} - H_{k+1}\hat{\mathbf{x}}_{k+1|k})$
State Covariance Update $P_{k+1|k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k}$
< Network Parameter Adaptation >
State Prediction $\hat{\mathbf{x}}_{k+2|k+1} = F_{k+1}\hat{\mathbf{x}}_{k+1|k+1}$
State Covariance Prediction $P_{k+2|k+1} = F_{k+1}P_{k+1|k+1}F_{k+1}^T + W_{k+1}Q_{k+1}W_{k+1}^T$
Future Event Detection Probability Computation
$\mathbf{p}^j_{k+2|k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \hat{\mathbf{x}}^j_{k+2|k+1}.$
$S^{(j,i)}_{k+2|k+1} = Pr(\mathbf{p}^j_{k+2|k+1} \in G(i)) = \int_{\mathbf{u}\in G(i)} S^j_{k+2|k+1}(\mathbf{u})d\mathbf{u}$
Network Parameter Determination for t_{k+2}

Fig. 3: The state transition diagram that describes the steps to determine a new network parameter value based on FEDP.

partitions and performing a Riemann sum:

$$S^{(j,i)}_{k+1|k} \approx \sum_{n=[0,m-1]} S^j_{k+1|k}(\mathbf{u}_n) f(\mathbf{u}_n).$$

where $f(\mathbf{u}_n)$ is the size of each cell in the discretization of the field-of-view.

4.4.2. Expected Time-of-arrival of Event. Since each node runs its own tracker and since there will be losses in packets containing indirect sensing information, the estimated object state and its associated covariance matrix are likely to be different at different nodes.

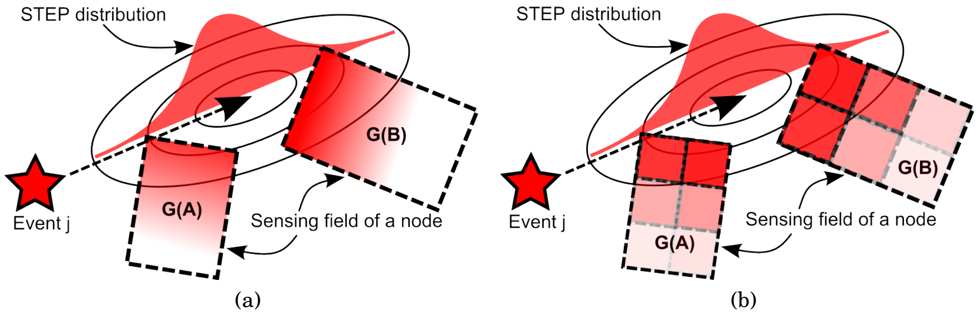


Fig. 4: STEP distribution for an event j and FEDP of Node A and B: **(a)** The FEDP value of Node A and B, $S^{(j,A)}_{k+1|k}$ and $S^{(j,B)}_{k+1|k}$, are the integration of STEP, $S^{j}_{k+1|k}(\mathbf{u})$, over $G(A)$ and $G(B)$, respectively. **(b)** The FEDP values can be approximated by a Riemann sum with m partitions: In this example, the sensing field of each node is divided into six partitions.

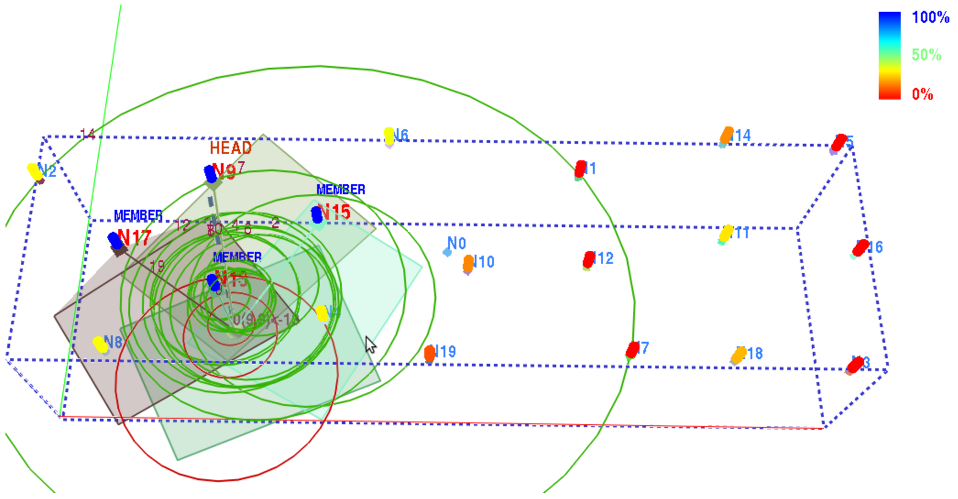


Fig. 5: Tracking by Kalman filter at each node. Small and large red circles indicate the current and the predicted positions of an object as estimated by the Kalman filter at the current cluster head. Small and large green circles are those estimated by the Kalman filter at each of non-cluster nodes. The color of the node indicates the probability that the object will enter its field-of-view in the near future (4 seconds).

Figure 5 illustrates this effect where the Kalman filter estimates of the current object position and the predicted position made at the cluster head are shown as red ellipses and the estimates made at other nodes (including those that are not part of a cluster) are shown as green ellipses. As we can see in Figure 5, nodes 11, 14, and 18 are more distant to the current object position than nodes 12 and 10. Due to the disparity in Kalman filter estimates at the different nodes, however, nodes 11, 14, and 18 measure a smaller distance to the target than nodes 12 and 10 in terms of the Mahalanobis distance (described as the color of a bar on top of each node). While such

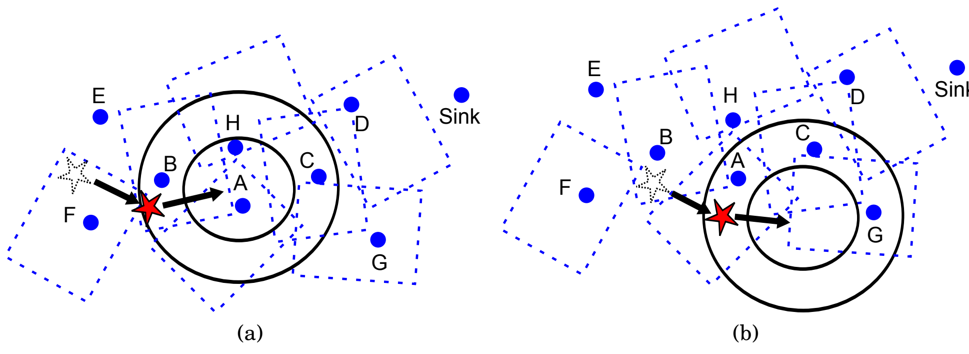


Fig. 6: The continued depiction of a WCN engaged in tracking a moving object at two subsequent time instants. The red star indicates an event and the black solid arrows its moving direction. The regions divided by black solid circles indicate examples of a contour map of the STEP of the event predicted by a node.

disparity is not evident in the predicted position itself, it is significant in its associated uncertainties (represented as the size of the green ellipsoids).

Given the estimate of the predicted position at time t_{k+1} as the mean of the STEP distribution, then a node can estimate the expected time-of-arrival of the event (EToE) since the node is aware of the speed of the object and the distance to the future position of the object. However, depending on the process noise in the tracker, in general, we can expect to see fluctuations in the velocity state. So, instead of directly using the current velocity state, therefore, one could employ a method to measure the mobility on a longer term basis in order to smooth out the fluctuations. How such smoothing should be carried out, would vary from target to target. For example, the noise properties related to pedestrian movements are obviously very different from the noise properties related to the movement of automobiles. A node must therefore observe a target before deciding on the best approach to use for the smoothing.

An understanding of the long-term mobility properties of a target object can be carried out by employing a circular buffer of size $N_{mobility}$ that stores the most recent $N_{mobility}$ estimated speeds of the object whenever the tracker is updated and averaging them. That yields a time-window-based average speed of the object. Note that the size of the circular buffer, $N_{mobility}$, should not be too long in case that the mobility of an object may change over time. Yet the size should be set large enough to reveal its typical mobility.

5. PREDICTIVE DUTY CYCLE ADAPTATION

While the tracker module keeps track of a target object, we would want the adaptation of the network parameters to take place in the individual protocol layers connected to the tracker module. As an example of such optimization, in this section we present how such adaptation can be carried out in the MAC layer with regard to the duty cycle at each node on the basis of FEDP and EToE. As we will show, the predictive duty cycle adaptation (PDCA) strategy we present for the MAC layer adjusts the duty cycle of a node in advance according to the FEDP or EToE before the target object is visible to the camera at the node. PDCA enables a node to quickly ready for the anticipated high traffic.

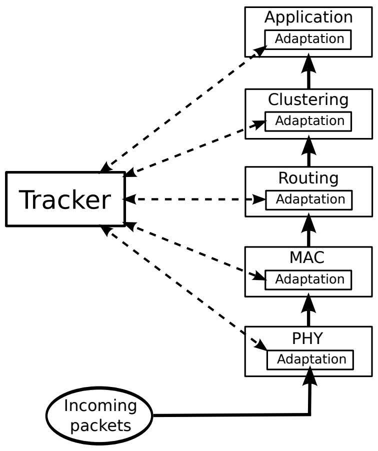


Fig. 7: Overall system architecture of PDCA: The tracker module is implemented separately, not belonging to any layer within the protocol stack. The adaptation modules reside in individual layers since each of them is responsible for adapting different network parameters in different layers. The tracker and adaptation modules are connected to each other and the whole system can be implemented as an add-on to an existing network system as the figure depicts.

5.1. Overall System Architecture

The proposed PDCA framework can be applied to any network systems with synchronous MAC protocols where a sender and its potential receivers share the same active periods. Figure 7 shows how PDCA can be added on to an existing network system. The protocol stacks in a typical network system have a single-path packet flow from the physical layer at the bottom to the top application layer. Equipped with PDCA as an add-on, a cross-layer interaction between all of the layers in the protocol stack via tracker module is now made possible. More specifically, each of the network layers that are connected to the tracker module feeds event-related information into the tracker so that the tracker can update the state of the event/object. At the same time, each layer is able to consult with the tracker about the current state of the event/object and if necessary adapt its network parameters. As a result, a PDCA-enabled network can achieve high energy efficiency and tracking performance. Such synergy is created when each node in the camera network performs both object tracking and network parameter adaptation simultaneously.

5.2. Determining an Appropriate Value for the Duty Cycle

Once the FEDP or EToE is computed in the tracker module at a node, each node chooses a value for its duty cycle in the MAC layer. A higher FEDP or a lower EToE would cause a node to choose a larger value for the duty cycle. Let's say we allow for N different levels of the duty cycle, d_0, d_1, ..., d_{N-1}, with d_{N-1} being the highest. Let d_c be the current value for the duty cycle level. Whenever a STEP update occurs at a node, the node computes its new FEDP or EToE and accordingly a new duty cycle level d_m. If $d_m \neq d_c$, then the node schedules a change of duty cycle to d_m and adopts the new schedule based on the new duty cycle level d_m. Subsequently, the node broadcasts this fact to its neighbors so that they can be aware of the updated communication schedule at the sender.

Symbol	Parameter
N	The maximum number of duty cycle level
T_n	The frame length of the base (lowest) duty cycle level d_n
M	The base of the exponentially varying frame length
t_c	The current time
t_{cf}	The time when the current frame started
t_{bf}	The time when the previous base frame started
t_{nf}	The time when the next frame is scheduled to start

Table I: Parameters and symbols used

Consider an example illustrated in Figure 6 where a target is being tracked and its future position is being predicted. The circles (in general, these will be ellipses) represent the equiprobable contours of the STEP distribution. If the target, which was initially detected by node F, moves to the sensing field of node B, thereby triggering packet transmissions from node B, as shown in Figure 6(a), then the EEN bit will be set for all the packets transmitted by node B, informing node A and H of the detection of the target at B. Upon the reception of a packet from node B, a Kalman filter in both node A and H will be created, initialized, and updated due to this indirectly sensed measurement. Nodes in the neighborhood of node B will then compute the current STEP $S^j_{k+1|k}(\mathbf{u})$ and FEDP $S^{(j,i)}_{k+1|k}$ with respect to their sensing fields. Consider for example the actions taken by node A: The STEP computed by node A based on the detected packets from node B is illustrated in Figure 6(a). At this moment, node A predicts that it is highly likely that the event will be detected in the next measurement due to a high FEDP value $S^{(j,A)}_{k+1|k}$, and consequently schedules to alter its duty cycle to the highest level d_{N-1} at t_{k+1}. At a subsequent time instant, shown in Figure 6(b), node A acquires an observation of the target and computes a new STEP and FEDP. At this point, node A realizes that the target is moving away from its sensing field, i.e., the new FEDP indicates that it would not need the highest duty cycle level by the time it acquires the next measurement. Note that since each node computes its own STEP independently based on not only its own measurements but also the detected packets from its neighbors, the STEP estimated at each node can be slightly different. Note also that since the sensing field of node A is closest to the center of the predicted event position in Figure 6 (a), its FEDP would be the highest among the nodes, causing it to have the highest duty cycle value d_{N-1}, while the other nodes would have relatively smaller FEDP values, resulting in adopting the same or lower duty cycle values.

In contrast with the enhanced traffic levels generated by the appearance of a target in the field-of-view of a node, the disappearance of the target can only be inferred by the absence of packets with EEN set for a period of time. This translates into a *soft state* approach for duty cycle adaptation. That is, we set a timeout period whenever a duty cycle modification occurs. Upon the expiration of the timeout period, we assume the target has left the augmented sensing field of the node, and the duty cycle is reset to the lowest level d_0. This soft state approach also prevents a node from changing its duty cycle too frequently. The duty cycle adaptation procedure is summarized in Algorithm 3. The definitions of parameters used are summarized in Table I.

5.3. Exponential Frame Length Adjustment

After a target is considered detected, directly or indirectly, and the corresponding duty cycle ascertained as described previously, the system next calculates the *frame length*, which is the sum of the on-time and the off-time for the radios. Any changes to the frame length are carried out by adjusting it exponentially. Let T_c be the current frame

ALGORITHM 3: Duty cycle adaptation

```
repeat
    if an event exists within the sensing field then                                  ▷ Direct sensing
        Schedule to adopt the highest duty cycle level $d_N$ at the earliest time slot;
    else if current time is almost scheduled time then                                ▷ Indirect sensing
        Compute STEP $S^j_{k+1|k}(\mathbf{u})$ for event $j$ and FEDP $S^{(j,i)}_{k+1|k}$ for a node $i$ ;
        Determine its proper duty cycle level $d_m$ based on FEDP ;
        Schedule/reschedule to adopt the new duty cycle level $d_m$ at $t_{k+1}$;
    if $d_c \neq d_m$ then
        Set $d_m$ to be the current duty cycle level with a timeout;
    else                                                                              ▷ $d_c == d_m$
        Refresh timeout of $d_c$;
    if a new schedule is adopted then
        Broadcast the new schedule to neighboring nodes;
until Node is running;
```

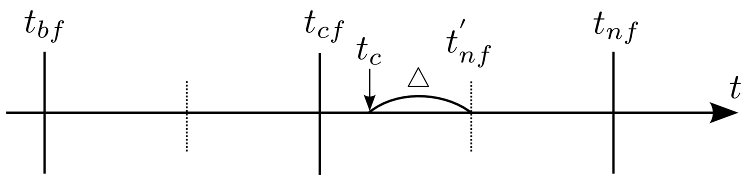


Fig. 8: Timeline for adopting a schedule with a higher duty cycle.

length corresponding to the duty cycle value of d_c, T_0 the frame length for d_0, the smallest value for the duty cycle, and M the base of an exponentially varying frame length. Then, T_c is one of

$$T_n = \frac{T_0}{M^n},$$

where $n \in \{0, ..., N-1\}$, and $M \in \mathbb{N}^*$. Note that in DSMAC and AMAC, M is always set to two, whereas in the proposed PDCA scheme it could be any non-negative integer number. If M is set to either 2 or 3, the frame length would change by doubling or tripling the interval between the starting time of the active period of two consecutive frames. For some MAC protocols such as TMAC, the length of the active and sleep period changes dynamically according to the current network conditions while the frame length remains fixed. Our PDCA scheme dynamically changes the length of the frame itself, regardless of whether the active and sleep periods change or not in a given frame length.

This exponentially varying adaptive frame method guarantees that any pair of nodes is able to communicate within a shared active period even if the nodes operate at different duty cycles. Suppose, for example, that two nodes i_n and i_m operate at duty cycle levels d_n and d_m, and that $n < m$. Suppose both nodes are initially active at time t_0. Node i_n has wake up times at $t_0 + kT_n$ where $k \in \mathbb{N}$, and node i_m at $t_0 + lT_m$ where $l \in \mathbb{N}$. Hence, whenever $\frac{l}{k} = M^{m-n}$, the active periods of both nodes will coincide. As a consequence, every node in the network is able to communicate with its immediate neighbors at least during the active periods of the *base frames*, which correspond to the frames given by the lowest possible duty cycle level d_0.

5.4. Adopting A New Schedule

After a node ascertains that its duty cycle must be changed to a new value — let's call it d_m — it is necessary to define mechanisms that allow for the communication schedule at the node to be modified without breaking synchronization with the neighboring nodes. In order to not cause a break in the synchronization, any modifications to this schedule must take place at the beginning of a frame. Let t_c denote the time instant when a node figures out that its new duty cycle should be d_m. The node must next determine when to start the new communication schedule with the new duty cycle.

Let t_{cf} be the time that the current frame started, t_{bf} the time when the previous base frame started, and t_{nf} the time when the next frame will start according to the current schedule. If a node decides to change its duty cycle to a different level d_m at time t_c , it schedules the beginning of the next frame to $t^{'}_{nf} = t_c + \triangle$, where $\triangle$, the residual time to the beginning of the next frame, is given by

$$\Delta = min\left[U\left(\frac{T_0}{M^{d_m}}\right) - (t_c - t_{bf})\right], \tag{4}$$

subject to $\Delta > 0$, where $U \in \{1, \ldots, M^{d_m}\}$. Since all the parameters in Eq. (4) are known and M^{d_m} is relatively small, this minimization problem can be solved quickly by simply searching over all the possible values of U.

Consider for example the timeline shown in Figure 8. In this figure, solid vertical lines illustrate the beginning of the active periods of the current schedule, and the dashed vertical lines show the beginning of the active periods of the new schedule to be adopted. Suppose that the parameters in this case are $M = 2$, $N = 5$, and the current duty cycle level of the node is $d_c = 1$. At time t_c, the node decides to increase its duty cycle to $d_m = 2$. Then, the parameter U will be chosen as the minimum between 1 and $M^{d_m} = 2^2$ that satisfies $\Delta > 0$. Since $t_c - t_{bf} > \frac{T_1}{2}$ in this example, it turns out that U is 3. Next, Δ is computed by setting $U = 3$ in Eq. (4), and the beginning time of the next frame $t^{'}_{nf}$ is rescheduled accordingly.

5.5. Communications Among Heterogeneous Schedules

A pair of synchronized nodes with different duty cycles can communicate with each other successfully only if they share a common active period. This overlap between the active periods at two different nodes is determined by the node with the shorter duty cycle.

In general, when a node attempts to send a packet to another node, it is not trivial to know whether the intended recipient is active or not when the nodes are allowed to have different radio schedules. In existing adaptive MAC protocols, the sender just tries to transmit a packet, hoping that the receiver is active. If ACK is not received, then the sender may try multiple retransmissions of the same packet in the same or the next active period. Such a trial-and-error approach obviously incurs additional overhead in terms of energy, traffic, and latency.

In order to overcome these limitations, we employ a novel frame numbering strategy that provides to the transmitting node an assurance that the receiving node is active. This completely eliminates the overhead associated with the trial-and-error approach yet at the cost of broadcasting a SYNC packet whenever a node changes its duty cycle to notify its neighbors of the change. The main idea is to assign a sequence of integer numbers to each frame that indicates the position of the current frame with respect to the base frame in such a way that the assigned frame number is *consistent* for all the neighboring nodes that share the same frame although they may have different duty cycles. Recall that the base frame length T_0 is the length of a frame when the duty cycle is at the lowest level d_0. Suppose node A is operating at the highest duty

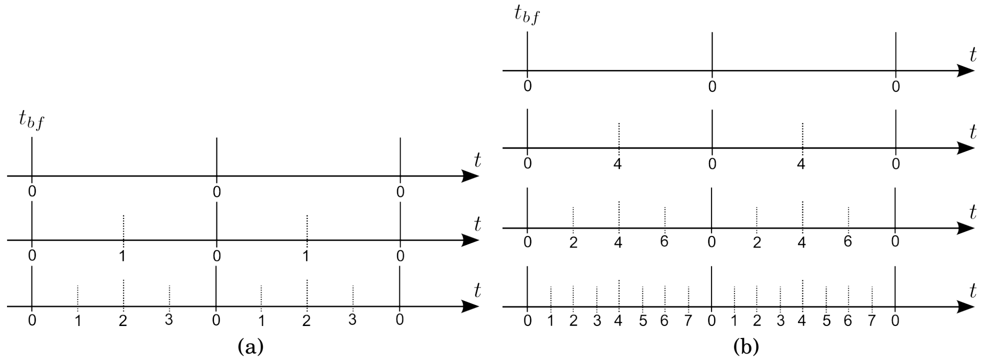


Fig. 9: (a) An example of inconsistent frame numbering, when $M = 2$ and $N = 3$. (b) An example of consistent frame numbering, when $M = 2$ and $N = 4$. From the top timeline, d_c corresponds to 0, 1, 2, and 3, respectively.

cycle level d_{N-1}, thus having $M^{d_{N-1}}$ frames within one base frame length. In this case, we can simply number each frame consecutively from 0 to $M^{d_{N-1}} - 1$. Now suppose a neighboring node B is operating at a lower duty cycle level d_c, $c < N - 1$, and thus can only have $M^{(d_{N-1}-d_c)}$ frames in one base frame length. If we again consecutively number these frames from 0 to $M^{(d_{N-1}-d_c)} - 1$, the frame number of the frames that node A and B share will be different at node A and B, resulting in inconsistency in frame numbering as shown in Figure 9(a). To avoid such inconsistency, we use the following simple equation to generate a sequence of frame numbers for a node with a particular duty cycle d_c:

$$f(i+1) = (f(i) + M^{(d_{N-1}-d_c)})\, mod\, M^{d_{N-1}} \tag{5}$$

where $f(i)$ is the frame number of the i-th frame and $f(0) = 0$. With this consistent frame numbering strategy as shown in Figure 9(b), the active periods of a node and the active periods of a neighboring node operating at a duty cycle d_c^{peer} are *guaranteed* to overlap whenever the following condition is true:

$$f(i)\, mod\, (M^{d_{N-1}-d_c^{peer}}) == 0. \tag{6}$$

For successful unicast communications, the node should transmit a packet to its neighbor only in such frames.

While the strategy presented above solves the problem of unicast communications between two nodes with different schedules, we still have the issue of broadcast communications among such nodes. For broadcast communications, multiple transmission policies are at out disposal: One could restrict the communications so that a node can broadcast a packet only if all of its neighbors can receive it, that is, only if Condition (6) is true for *all* of its neighbors. This approach, evidently, incurs longer transmission delays for broadcast packets. On the other hand, it is also possible to broadcast messages as long as at least one neighbor is awake, that is, if Condition (6) is true for *at least one* neighbor. Although this may reduce the chance of neighbors receiving the packets, the PDCA scheme employs this approach since the nodes in the vicinity of an event are highly likely to have the same or even higher duty cycle, and the event-related information is usually delay-sensitive.

5.5.1. Neighbor Synchronization . All synchronous MAC protocols require that SYNC messages be exchanged by the nodes in order to maintain time synchronization. What

ALGORITHM 4: Support for fast delivery of routing packets related to an event of interest

```
$p_{in}(i) \triangleq$ a packet received from node $i$ via unicast    ▷ Intended recipient
$p_{out}(i) \triangleq$ a packet to be transmitted to node $i$
$isEventRouting \triangleq$ a flag indicating whether a node is part of a routing path of event-related packets
repeat
    if a packet $p_{in}(i)$ is detected with EEN or EERN bit set then
        Set $isEventRouting$ with a timeout;
        $d_m \leftarrow d_{routing}$ ;
        Schedule/reschedule to adopt the new duty cycle level $d_m$;
    if a packet $p_{out}(i)$ is ready then
        $p_{out}(i).EERN \leftarrow isEventRouting$;
    if timeout expires then
        Unset $isEventRouting$ ;
        $d_m \leftarrow d_0$ ;
        Schedule/reschedule to adopt the new duty cycle level $d_m$;
        Broadcast the new schedule to neighboring nodes;
until Node is running;
```

is placed in the SYNC packets depends on the MAC protocol and its synchronization policies. For our case, in order to enable PDCA, the SYNC packets also contain (1) the address of the schedule initiator; (2) the current duty cycle level d_c; (3) the residual time to the beginning of the next base frame; (4) the age of the current schedule; and (5) a 2-bit field for EEN and EERN that is discussed in the next section. The age of a schedule refers to the number of times that the schedule was broadcasted by the initiator in periodic exchanges. PDCA also requires that the SYNC packet be sent whenever a schedule change occurs at a node due to a change in the local duty cycle.

5.6. Fast Delivery of Event-related Packets

To reduce latency in the delivery of the packets containing event-based information back to the base station, the system must be able to identify the intermediate nodes along the routing path to be used for the delivery of such packets. As we previously discussed, nodes that detect events, directly or indirectly, set the EEN bit in the MAC header of the outgoing packets. In order to indicate that a node has been selected for routing event-related packets to the base station, we define the *Explicit Event-Routing Notification* (EERN) bit in the MAC header.

A node that is on a routing path increases its duty cycle to a pre-defined level $d_{routing}$ to minimize the end-to-end latency. For example, $d_{routing}$ could be set to the maximum duty cycle d_{N-1}. Consider, for example, the WCN shown in Figure 1(c). Since the nodes B, H, C, D, and Sink are along the routing path of the event-related packets, their duty cycle would be increased to $d_{routing}$ upon the reception of packets originated from nodes E or F.

The routing-path membership of a node is considered to be a soft state that must be refreshed periodically by the reception of packets with EERN set. If a node does not receive a routing packet within a specific period of time, it will reduce its duty cycle to the lowest level d_0. Duty cycle adaptation for routing event information is summarized in Algorithm 4. In the algorithm, the variable *isEventRouting* indicates whether a node is currently a part of a routing path.

6. PERFORMANCE EVALUATION

We now present two evaluations of the adaptive duty cycling approach presented in this paper. One is computer-simulation based and the other on an actual wireless camera network consisting of Imote2 nodes equipped with cameras.

Our evaluation is based on metrics that include Quality-of-Service (QoS) metrics designed specifically for wireless camera networks. Note that, in general, the definition of QoS depends on the intended application of a WSN. Bianchi et al. [Bianchi et al. 1996], for example, analyzed the throughput and access delay of the IEEE 802.11 MAC protocol as a function of various contention windows. Their QoS evaluation metrics were the prioritization capabilities of the several MAC operation modes, including network utilization, latency and throughput. He et al. [He et al. 2004] presented a novel way to achieve energy efficiency in a WSN for an object tracking system using a sentry-based power management. They claim that the precision in the location estimate and the latency in reporting an event to the base station are important QoS metrics for the specific application of tracking performance.

Our primary application space is target tracking with a WCN. Since this application requires that the nodes engage in collaborative processing of the sensed data for scene interpretation, the QoS metrics used must reflect this fact. The widely used performance metrics such as latency and throughput do not capture the unique properties of WCNs and thus are not sufficient for our evaluation. We therefore include application-level QoS metrics that were designed specifically for WCNs. These QoS metrics are intended to evaluate performance in data aggregation and clustering operations for collaborative processing in WCNs. Using a QoS metric for data aggregation (i.e., the average TIBPEA which will be reviewed in the next subsection) and three QoS metrics for clustering operations (which will be elaborated in Section 6.3) as well as energy efficiency (in terms of the average effective duty cycle), we conduct performance evaluation, first, with a large-scale simulation, and, then, on a real testbed based on Imote2 motes with cameras.

Before presenting the rest of the material in this evaluation, we recall that the PDCA framework is an add-on functionality for network systems with a synchronous MAC protocol. We have chosen the well-known synchronous MAC protocol known as TMAC [Van Dam and Langendoen 2003] as the basic MAC protocol for our experiments. We retrofit PDCA to TMAC for predictive duty cycling. We refer to this combination as P-TMAC. We compare the performance of P-TMAC with just TMAC to validate the predictive duty cycling approach presented in this paper. Note that our PDCA can be applied to different MAC protocols as long as they maintain a synchronous communication schedule among the nodes.

6.1. Quality-of-Service Metric

In Section 2.1, we showed the traffic patterns associated with typical cluster-based collaborative processing. Earlier, in [Shin et al. 2007], we established that a good QoS metric for such traffic patterns in WCN is the TIBPEA metric. (TIBPEA stands for Time Bounded Parameter Estimation Accuracy.) TIBPEA is based on the rationale that the dominant factor that contributes to the accuracy in cluster-based object tracking is the number of measurements available for each aggregation. TIBPEA for a cluster-based object tracking application is computed by the average percentage of the cluster members that successfully reply to the cluster head within a certain timeout period.

Consider again the WCN shown in Figure 1. Suppose an event occurs at Node B and it is elected as a cluster head while its one-hop neighbors such as Node E, F, H, and A are its cluster members as in Figure 6 (a). Then, when Node B transmits a

Tx range	$\sim 100meters$	SYNC	$22Bytes$
Tx power	$42.24mW$	RTS/CTS	$14Bytes$
Rx power	$38mW$	ACK	$14Bytes$
Sleep power	$15\mu W$	DATA	$44Bytes$
Idle power	$3mW$	Sim. time	$2400Sec.$

Table II: Summary of simulation parameters.

broadcast message to request measurements from its members, it expects to receive four measurements within a timeout. If Node B ends up receiving only three of them due to severe contention or whatever reason, then TIBPEA in this round would be $\frac{3}{4} = 0.75$. Suppose again the event occurs to Node H as in Figure 6 (a) and it is elected as a cluster head afterward. If it receives only two measurements from its members when it requested measurements, then TIBPEA would be computed as $\frac{2}{3} = 0.67$ since it has three cluster members such as Node A, B, and C. The network-wide average of TIBPEA values over time will then be used for performance evaluation.

6.2. Simulation-based Evaluation

While TMAC allows for active time adaptation, it does not allow for frame length adaptation, and, even more importantly, it does not allow for adaptation to be based on prediction. TMAC only reacts to the current network conditions by adapting the length of the active period. By applying the PDCA scheme to TMAC, the frame length also becomes dynamic, and duty cycle adjustments are carried out in a predictive manner. The result is better adaptation without any design conflict. TMAC modified in this manner will be called *P-TMAC*.

In TMAC, different nodes in the network may operate under different schedules because a node can randomly initiate its own schedule in the initialization stage if it does not receive any schedule for a certain period of time, which often results in multiple border nodes with different schedules. Since border nodes create severely unbalanced energy consumption in the network and introduce additional delays in routing, for simplicity, we employ a simple global scheduling scheme in the entire network solely for performance evaluation purposes. If a node that has a schedule receives a new schedule, then it adopts the schedule that was created earlier.

We evaluate P-TMAC in the context of target tracking using the Castalia simulator [Boulis 2007] which is based on OMNeT++. We simulate a network consisting of 200 TelosB nodes equipped with cameras hung randomly from the ceiling and pointing downwards. The nodes cover a $200m \times 200m$ area. The sensing range of each camera is a circle with a radius of $40m$. A randomly moving object is assumed to exist in the network during one third of the total simulation time.

We compare the performance between P-TMAC and TMAC. The base frame length of P-TMAC is set to $T = 1000ms$, its active period to $30ms$, and its frame length is allowed to vary among $N = 4$ levels, corresponding to T, $T/2$, $T/4$, and $T/8$, that is, $M = 2$. Since the active period remains constant, these frame lengths correspond to duty cycles of 3%, 6%, 12%, and 24%, respectively. To ensure a fair comparison, we evaluate TMAC operating at the same four duty cycles. In our experimental results, these different TMAC instances are identified as TMAC-3, TMAC-6, TMAC-12, and TMAC-24. The detailed parameters used in our evaluation are summarized in Table II.

We simulated two types of scenarios: In the first scenario, each node that detects an object directly reports that fact to a base station. In the second scenario, whenever a node detects an object, it creates a new cluster or it joins an existing cluster. The elected cluster head broadcasts a request message to its members to perform col-

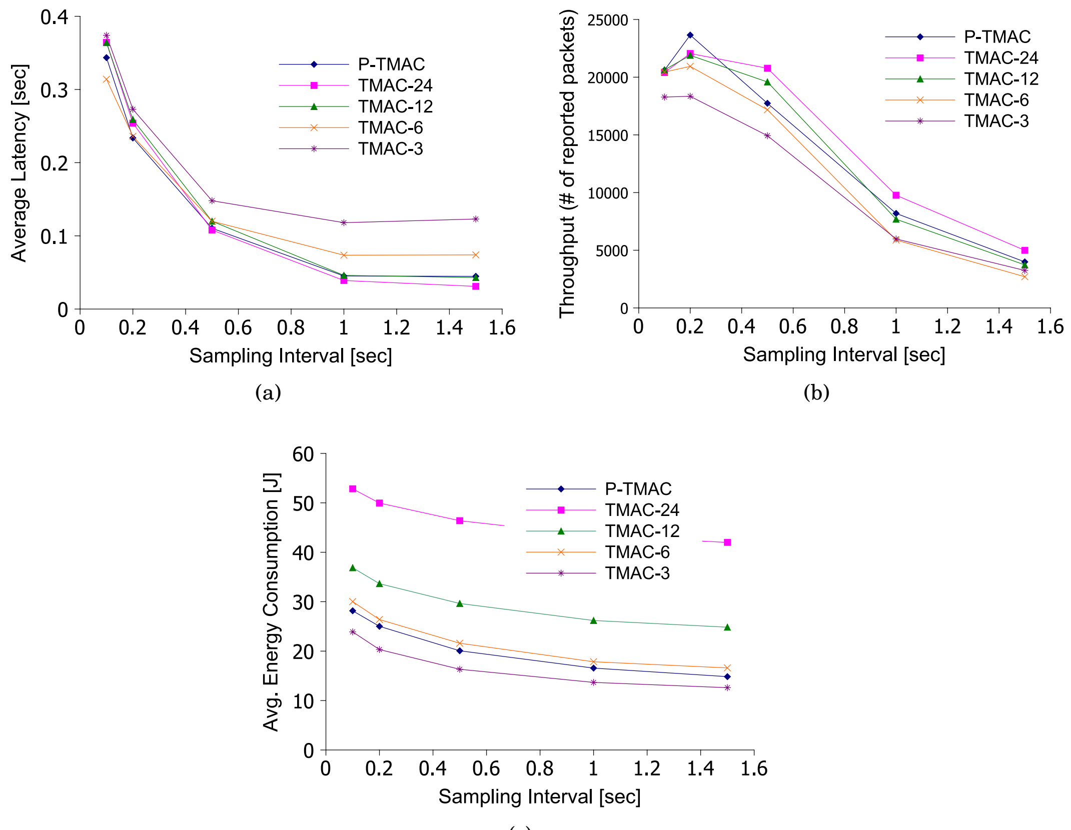


Fig. 10: Simulation results of network performance in terms of (a) latency, (b) throughput, and (c) energy consumption of P-TMAC and TMAC with four different duty cycles.

laborative sensing and processing, and each cluster member replies by unicasting its measurement. In the first scenario, we evaluate the performance of the MAC protocols in terms of a set of traditional metrics such as latency, throughput, and energy efficiency. The energy efficiency is evaluated based on the consumed energy only by the radio. To successfully capture the performance characteristics of the MAC protocols in the second scenario, we employ the aforementioned application-level QoS metric TIBPEA.

6.2.1. Individual Processing and Reporting Scenario. Because of the adaptive frame length design, P-TMAC is expected to show performances in between TMAC-3 and TMAC-24. Figure 10(a) shows that the latency of P-TMAC is comparable to that of TMAC-24 at different sampling intervals. Figure 10(b) shows the throughput evaluation results. Obviously, shorter sampling intervals entail higher packet rates.

To interpret this result, let us define the period from the time an object of interest enters the sensing field of a node to the time it leaves it as the *sensing round*. Let us also define the first packet transmitted during each sensing round as the *link initializing packet*. With the same object motion, higher sampling rate causes more packet generation per sensing round, resulting in a small proportion of link initializing pack-

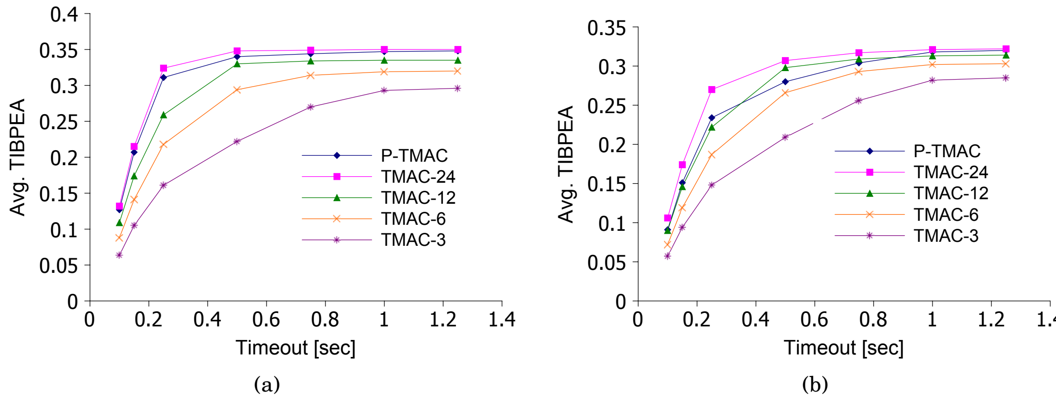


Fig. 11: Simulation results of the average TIBPEA with different average target speeds: (a) 6m/s and (b) 24m/s.

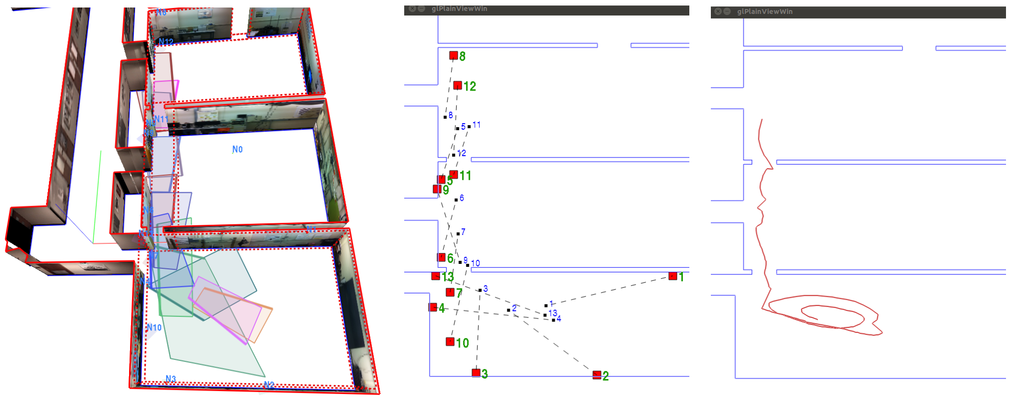
ets to the overall number of packets. TMAC is designed to work best when the rate of link initializing packets is low because of low sampling interval or slow object movement. As we can see in Figure 10(a), the average per-hop latency of TMAC-3 increases as the sampling interval increases while P-TMAC retains its performance similar to that of TMAC-24.

Providing performance that is comparable to TMAC-24 on the basis of latency and throughput, P-TMAC achieves an energy efficiency level between TMAC-3 and TMAC-6, as shown in Figure 10 (c). It implies that P-TMAC substantially improves the trade-off between energy and latency compared to TMAC.

6.2.2. Collaborative Processing and Reporting Scenario. When a node detects an object of interest in this scenario, it tries to collect its neighbors' measurements to obtain more in-depth understanding of the object by collaborative data processing. We conduct two sets of simulations with average target speeds of 6m/s and 24m/s. In each set, the average TIBPEA is measured with different timeout bounds. In all simulations, when the timeout bound is tight, the performance of P-TMAC is comparable to that of TMAC-24, as shown in Figure 11. When the timeout bound is loose, P-TMAC still shows better performance than TMAC-3 but worse than TMAC-24. This is caused by the inherent additional communication overhead of P-TMAC for broadcasting SYNC messages whenever a duty cycle adaptation occurs. Nonetheless, the superior performance of P-TMAC for delay-critical applications satisfies our design goal.

6.3. Evaluation Using Real Data on Imote2-based Testbed

With regard to performance evaluation with real data, we have evaluated PDCA on a testbed consisting of 13 Imote2 nodes that span three rooms, each roughly $20ft \times 20ft$, as shown in the 3D model of the rooms in Figure 12(a) and the plan view in Figure 12(b). It is important to note that this spatial layout is located in one of the oldest buildings on campus and that the rooms are separated by thick masonry walls with embedded wire meshing for reinforcement. So the usual formulas for the single-hop distance one may associate with the radio emanations from Imote2 nodes would not apply in this case. The linear distance between the node at one end (node 1 in Figure 12(b)) and the node at the other end (node 8 in the same figure) is approximately 80 ft. It takes two radio hops for the node at one end to communicate with the node at the other end. The cameras at the locations shown in the plan view in Figure 12(b)



(a) (b) (c)

Fig. 12: The RVL wireless camera network testbed used for this validation consists of 13 Imote2 motes with cameras deployed across three rooms: (a) A 3D model of the testbed with the sensing region of each camera depicted as a colored polygon on the floor; (b) A plan view of the testbed with the physical location of each camera drawn as a small red box and the center of its sensing range drawn as a small black box connected to the red box with a dotted line; and (c) An example of a mobile object's trajectory estimated by the testbed.

are oriented in such a way that it is safe to assume that as a target object travels in the area monitored by the 13 cameras, it will always be visible to more than two cameras at a time. Note that each of the cameras is calibrated. What that means is that each camera knows its position and orientation in a global frame of references. When tracking "flattish" objects, each camera can use its calibration parameters to calculate the center of mass of the object on the floor, assuming that the object is visible to the camera. Since such calculations are standard in computer vision [Hartley and Zisserman 2004], we will not go into them here. Figure 12(c) shows the track as computed by the network for an object piloted by a remote controller.

For the MAC layer in our experiments, the base M of the exponentially varying frame length is set to 2 while the maximum duty cycle level is also 2 (that is, N is set to 2 in Section 5.2), and the length of the active period of a frame is set to $300ms$ while the base frame length is $4000ms$. Therefore, the individual nodes in the network can have up to three different duty cycle levels at any given time. This translates into the maximum duty cycle being $300/(4000/2^2) = 30\%$ and the minimum duty cycle being $300/(4000/2^0) = 7.5\%$. With regard to the object tracked in these experiments, we used a toy vehicle that can be navigated with a handheld controller.

With regard to evaluating the effectiveness of the duty cycling achieved, we use the TIBPEA QoS metric and energy efficiency. Our choice of TIBPEA is dictated by the fact that the more traditional network-level performance metrics such as end-to-end latency and throughput do not capture the performance of a WCN that must engage in collaborative processing of sensed data. As we will point out later in this section, even TIBPEA has certain limitations with regard to capturing the true benefits of using our adaptive approach to duty cycling. Said another way, while the information conveyed by TIBPEA is necessary, it is not sufficient. Therefore, in addition to showing performance evaluation with TIBPEA, we will present comparative results using

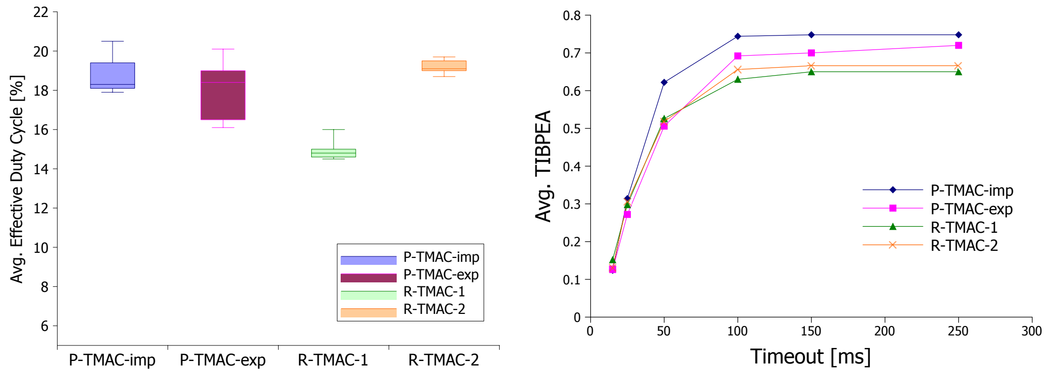


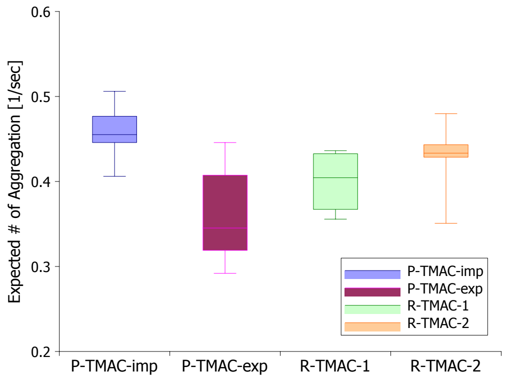
Fig. 13: Performance comparison in terms of (a) energy efficiency measured as the average effective duty cycle and (b) the average TIBPEA with varying timeouts.

other criteria. As we explain later, the shortcoming of TIBPEA is not relevant to our simulation results.

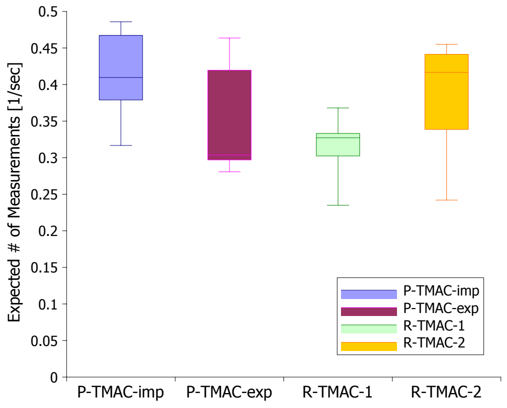
Our experimental evaluation is a 4-way comparison between the following: (1) PDCA with the EEN bit set in the MAC header; (2) PDCA without the EEN bit in the MAC header, but now the cluster head must broadcast the state information in separate packets periodically; (3) Reactive duty cycling in which the duty cycle at a node is modified only when the node directly sees the target; and (4) The same reactive duty cycling but with a higher minimum duty cycle. In our presentation of the results, we refer to the first case as P-TMAC-imp, where “imp” stands for implicit notification of event information using the EEN bit, and the second case as P-TMAC-exp, where “exp” stands for explicit broadcast of the state by the cluster head. We refer to the third and fourth cases as R-TMAC-1 and R-TMAC-2, respectively, where ’R’ stands for “reactive”.

With regard to the two reactive schemes in our comparative study, R-TMAC-1 and R-TMAC-2, the minimum duty cycle of R-TMAC-2 is set at twice the level of the other three approaches while maintaining its maximum duty cycle to be the same as others. Consequently R-TMAC-2 has only two levels of duty cycle while the other three approaches have three. R-TMAC-2 can therefore be expected to yield high performance in tracking but at the cost of low energy efficiency. By the same token, we can expect R-TMAC-1 to yield high energy efficiency and low object tracking performance. Including these two reactive approaches in our comparative study allows us to demonstrate the performance gain of the proposed predictive method in terms of both application-level performance (i.e., tracking) and energy efficiency.

Figure 13 shows the performance comparison between P-TMAC-imp, P-TMAC-exp, R-TMAC-1, and R-TMAC-2 in terms of energy efficiency measured as *the average effective duty cycle,* as shown in Figure 13(a), and *the average TIBPEA,* as shown in Figure 13(b), with varying timeouts. The average effective duty cycle is computed by the total active duration of the radios divided by the total running time. As expected, R-TMAC-1 consumes less energy than the other three approaches. Note that R-TMAC-2 has a higher minimum duty cycle than others that would result in higher overall energy consumption in a large-scale network where only a small subset of nodes are expected to adapt their duty cycle as an object is tracked while the rest are at the minimum duty cycle. Since our testbed consists of only 13 camera nodes, the energy consumption caused by a larger duty cycle at the nodes in the vicinity of an object tends to



(a)



(b)

Fig. 14: Performance comparisons in terms of (a) the expected number of data aggregations per second at the cluster heads for Kalman updating of the object position; and (b) the expected number of measurements per second that are reported to the cluster heads during tracking.

make a larger impact on the overall energy efficiency than would be the case in a larger network in which a smaller fraction of the nodes would be engaged in actual object tracking at any given time. Figure 13(b) shows that P-TMAC-imp outperforms the other three approaches in terms of the average TIBPEA. Figure 13(b) establishes conclusively that a reactive approach with a duty cycle twice as long as the other reactive approach does not yield a commensurate increase in the performance as measured by the average TIBPEA. With regard to a comparison of the predictive versus the reactive approaches in Figure 13(b), on the basis of the average TIBPEA results shown in Figure 13(b), it does not appear that the predictive approaches are overwhelmingly superior to the reactive approaches. Obviously, the predictive approaches are no worse than the reactive approaches. In what follows, we will explain that the story told by the performance curves in Figure 13(b) for the predictive approaches vis-a-vis the reactive approaches is incomplete. In other words, those curves are necessary but not sufficient for fully characterizing the network performance that is achieved with predictive approaches — especially the predictive approach presented in this paper.

TIBPEA assumes that the clusters have *already* elected their cluster heads for the parameters it measures — it measures the rate at which the cluster members succeed in communicating their measurements to the cluster head given a certain timeout. Although a necessary measure of the performance of a camera network, TIBPEA does not measure the main reason for predictive duty cycling, which is the ability to increase the duty cycle at a node *in advance* of the object actually arriving in its field-of-view for agile handling of upcoming traffic. As mentioned previously, such advance alteration of the network parameters allows for various clustering operations (for example, cluster formation, propagation, fragmentation, coalescence, etc.) to be executed smoothly while tracking objects. Therefore, to appreciate the full power of a predictive duty cycling approach such as ours, we also need to evaluate it from the standpoint of the

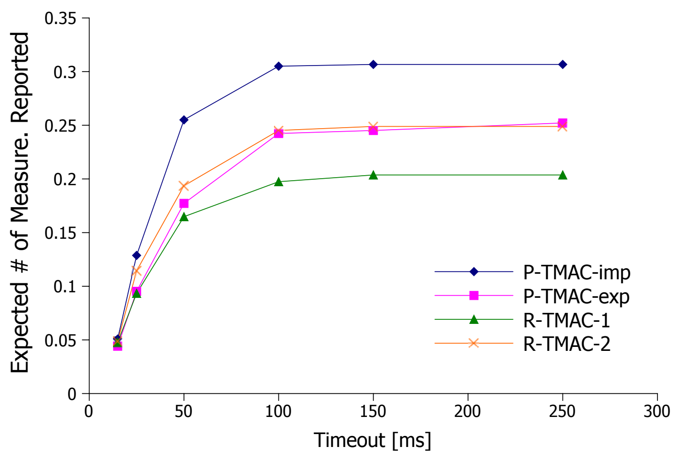


Fig. 15: Performance comparisons in terms of the number of measurements that a cluster head expects to receive from its cluster members within a varying timeout during object tracking. The data shown is averaged over all the clusters formed during tracking.

efficiency with which the clustering operations can be carried out. Average TIBPEA does not measure those effects in a network.[2]

To evaluate how well the clustering operation is supported by the MAC layer protocol, we have measured (1) *the expected number of data aggregations* per second at the cluster heads for Kalman-filter based updating of the object position; (2) *the maximum expected number of measurements* per second that can be reported to the cluster heads during tracking; and (3) t*he expected number of measurements* per second that are actually reported to the cluster head *within a varying timeout*. By experimental design, a cluster head polls the cluster members every $950ms$ for any data they may have to report. This implies that a data aggregation at a cluster head can be carried out every $950ms$. Figure 14(a) shows the expected number of data aggregations per second at the cluster heads for the four different approaches. These numbers are calculated by dividing the total number of data aggregations made at all the cluster heads by the total time during which the object was present within the sensing coverage of the network. Figure 14(b) then shows the maximum expected number of measurements per second that can be reported to the cluster head. More specifically, the results shown in Figure 14(b) are computed as the total number of measurements generated at the cluster members to be reported to their cluster heads divided by the total time when the object was present in the network. Figure 15 then shows how many of the generated measurements at the cluster members would be successfully reported to the cluster heads within a certain timeout. Since in the cluster-based distributed object tracking, more measurements at a higher data aggregation rate at the cluster head would yield a higher tracking accuracy with a lower error bound, we believe that the results shown in Figure 15 strongly demonstrate how well our predictive MAC protocol supports the distributed object tracking application. In both Figures 14(a) and (b), it is evident that our proposed P-TMAC-imp outperforms R-TMAC-1 while being com-

[2]In the comparative results in Section 6.2 where we have only used the average TIBPEA metric, note that those involved only one adaptive approach — our predictive approach. In this section, however, all the duty cycling approaches we are comparing are adaptive — even the reactive ones — with different adaptation strategies that have a bearing on the efficiency of clustering operations.

parable to R-TMAC-2 which incurs a higher energy cost. In Figure 15, P-TMAC-imp significantly outperforms both reactive approaches in terms of the expected number of measurements within a varying timeout. These results demonstrate that our proposed predictive approach better supports the dynamic changes in event-driven network operations, allowing for more frequent data aggregations with more measurements made available to the cluster heads.

7. CONCLUSION

We have presented a predictive duty cycle adaptation scheme (PDCA) suitable for wireless camera networks. The adaptation at a node takes place on the basis of the probability of object arrival at the node. This probability is estimated using a Kalman filter that takes into account both the direct measurements of the object position, when the object can be seen by the camera at the node, and on the basis of the indirect measurements of the object's position as conveyed by the EEN bit in the MAC header. Our simulation results confirm that our approach outperforms TMAC in terms of the TIB-PEA metric and presents a better tradeoff between energy efficiency and throughput or latency. We found that our approach is superior to others especially when the mobility of an event is large and when the latency requirement is strict. These results were further confirmed by our experiments involving a network of Imote2 nodes.

References

ANWANDER, M., WAGENKNECHT, G., BRAUN, T., AND DOLFUS, K. 2010. BEAM: A Burst-aware Energy-efficient Adaptive MAC protocol for Wireless Sensor Networks. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*. 195 –202.

BIANCHI, G., FRATTA, L., AND OLIVERI, M. 1996. Performance evaluation and enhancement of the csma/ca mac protocol for 802.11 wireless lans. In *Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on*. Vol. 2. 392 –396 vol.2.

BOULIS, A. 2007. Castalia: revealing pitfalls in designing distributed algorithms in WSN. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM, New York, NY, USA, 407–408.

BUETTNER, M., YEE, G. V., ANDERSON, E., AND HAN, R. 2006. X-MAC: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*. SenSys '06. ACM, New York, NY, USA, 307–320.

CERPA, A., ELSON, J., ESTRIN, D., GIROD, L., HAMILTON, M., AND ZHAO, J. 2001. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev. 31*, 20–41.

CHARFI, Y., WAKAMIYA, N., AND MURATA, M. 2009. Challenging issues in visual sensor networks. *Wireless Communications, IEEE 16,* 2, 44 –49.

COATES, M. 2004. Distributed particle filters for sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*. IPSN '04. ACM, New York, NY, USA, 99–107.

DOUCET, A., GODSILL, S., AND ANDRIEU, C. 2000. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing 10*, 197–208. 10.1023/A:1008935410038.

EL-HOIYDI, A. AND DECOTIGNIE, J. 2004. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. *Lecture Notes in Computer Science 3121*, 18–31.

FARINA, A., GOLINO, G., CAPPONI, A., AND PILOTTO, C. 2005. Surveillance by means of a random sensor network: a heterogeneous sensor approach. In *Information Fusion, 2005 8th International Conference on*. Vol. 2. 8 pp.

GUI, C. AND MOHAPATRA, P. 2004. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*. MobiCom '04. ACM, New York, NY, USA, 129–143.

HARTLEY, R. I. AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision* Second Ed. Cambridge University Press, ISBN: 0521540518.

HE, T., KRISHNAMURTHY, S., STANKOVIC, J. A., ABDELZAHER, T., LUO, L., STOLERU, R., YAN, T., GU, L., HUI, J., AND KROGH, B. 2004. Energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. MobiSys '04. ACM, New York, NY, USA, 270–283.

HURNI, P. AND BRAUN, T. 2010. MaxMAC: A Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks. In *Wireless Sensor Networks*. Lecture Notes in Computer Science Series, vol. 5970. Springer Berlin Heidelberg, 289–305.

JURDAK, R., BALDI, P., AND LOPES, C. V. 2007. Adaptive low power listening for wireless sensor networks. *IEEE Transactions on Mobile Computing 6*, 988–1004.

KUSY, B., DUTTA, P., LEVIS, P., MAROTI, M., LEDECZI, A., AND CULLER, D. 2006. Elapsed time on arrival; a simple and versatile primitive for canonical time synchronisation services. *Int. J. Ad Hoc Ubiquitous Comput. 1,* 4, 239–251.

LIN, P., QIAO, C., AND WANG, X. 2004. Medium access control with a dynamic duty cycle for sensor networks. In *2004 IEEE Wireless Communications and Networking Conference, 2004. WCNC*. Vol. 3.

LIU, S., FAN, K.-W., AND SINHA, P. 2009. CMAC: An energy-efficient mac layer protocol using convergent packet forwarding for wireless sensor networks. *ACM Trans. Sen. Netw. 5,* 4, 1–34.

MAINLAND, G., PARKES, D. C., AND WELSH, M. 2005. Decentralized, adaptive resource allocation for sensor networks. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*. NSDI'05. USENIX Association, Berkeley, CA, USA, 315–328.

MEDEIROS, H., PARK, J., AND KAK, A. C. 2007. A Light-Weight Event-Driven Protocol for Sensor Clustering in Wireless Camera Networks. In *First ACM/IEEE International Conference on Distributed Smart Cameras, 2007. ICDSC '07*. 203–210.

MIR, Z. H., KO, Y.-B., PARK, S., AND PYO, C. S. 2010. Ec-mac: A cross-layer communication protocol for dynamic collaboration in sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*. 1 –6.

NAM, Y., LEE, H., JUNG, H., KWON, T., AND CHOI, Y. 2006. An adaptive MAC (A-MAC) protocol guaranteeing network lifetime for wireless sensor networks. In *Proceedings of 12th European Wireless Conference (EW 2006), Athens, Greece*.

POLASTRE, J., HILL, J., AND CULLER, D. 2004. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, New York, NY, USA, 95–107.

RHEE, I., WARRIER, A., AIA, M., MIN, J., AND SICHITIU, M. L. 2008. Z-MAC: a hybrid MAC for wireless sensor networks. *IEEE/ACM Trans. Netw. 16,* 3, 511–524.

SHIN, P. J., PARK, J., AND KAK, A. C. 2007. A QoS Evaluation Testbed for MAC Protocols for Wireless Camera Networks. In First ACM/IEEE International Conference on Distributed Smart Cameras, 2007. ICDSC '07. *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, 235–242.

SORO, S. AND HEINZELMAN, W. 2009. A survey of visual sensor networks. *Advances in Multimedia 2009*.

TSENG, V. S. AND LU, E. H.-C. 2009. Energy-efficient real-time object tracking in multi-level sensor networks by mining and predicting movement patterns. *Journal of Systems and Software 82,* 4, 697 – 706. Special Issue: Selected papers from the 2008 IEEE Conference on Software Engineering Education and Training (CSEET08).

VAN DAM, T. AND LANGENDOEN, K. 2003. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM New York, NY, USA, 171–180.

WELCH, G. AND BISHOP, G. 1995. An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Chapel Hill, NC*.

XU, Y., HEIDEMANN, J., AND ESTRIN, D. 2001. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*. MobiCom '01. ACM, New York, NY, USA, 70–84.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*. Vol. 3.

ZHANG, W. AND CAO, G. 2004. Dctc: dynamic convoy tree-based collaboration for target tracking in sensor networks. *Wireless Communications, IEEE Transactions on 3,* 5, 1689 – 1701.