Fall 2013

Predictive Duty Cycling of Radios and Cameras using Augmented Sensing in Wireless Camera Networks

Joonhwa Shin
Purdue University

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Joon Hwa Shin

Entitled
Predictive Duty Cycling of Radios and Cameras using Augmented Sensing in Wireless Camera Networks

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

AVINASH C. KAK, Co-Chair	
Chair	
JOHNNY PARK, Co-Chair	
XIAOJUN LIN	
SAURABH BAGCHI	

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): JOHNNY PARK, Co-Chair

Approved by: M. R. Melloch 9/2/13
Head of the Graduate Program Date

PREDICTIVE DUTY CYCLING OF RADIOS AND CAMERAS USING AUGMENTED SENSING IN WIRELESS CAMERA NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Joonhwa Shin

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2013

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

Page

LIST OF FIGURES

ABSTRACT

Shin, Joonhwa Ph.D., Purdue University, December 2013. Predictive Duty Cycling of Radios and Cameras using Augmented Sensing in Wireless Camera Networks. Major Professors: Avinash C. Kak and Johnny Park.

Energy efficiency dominates practically every aspect of the design of wireless camera networks (WCNs), and duty cycling of radios and cameras is an important tool for achieving high energy efficiencies. However, duty cycling in WCNs is made complex by the camera nodes having to anticipate the arrival of the objects in their field-of-view. What adds to this complexity is the fact that radio duty cycling and camera duty cycling are tightly coupled notions in WCNs.

In this dissertation, we present a predictive framework to provide camera nodes with an ability to anticipate the arrival of an object in the field-of-view of their cameras. This allows a predictive adaption of network parameters simultaneously in multiple layers. Such anticipatory approach is made possible by enabling each camera node in the network to track an object beyond its direct sensing range and to adapt network parameters in multiple layers before the arrival of the object in its sensing range. The proposed framework exploits a single spare bit in the MAC header of the 802.15.4 protocol for creating this beyond-the-sensing-rage capability for the camera nodes. In this manner, our proposed approach for notifying the nodes about the current state of the object location entails no additional communication overhead. Our experimental evaluations based on large-scale simulations as well as an Imote2-based wireless camera network demonstrate that the proposed predictive adaptation approach, while providing comparable application-level performance, significantly reduces energy consumption compared to the approaches addressing only a single layer adaptation or those with reactive adaptation.

1. INTRODUCTION

Central to the design of a wireless sensor network (WSN) is the need for high energy efficiency and it is also true for wireless camera networks (WCNs) where the complexity of sensing and data processing is much higher. Obviously every hardware component and functional software module consumes energy yet at different rates. 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 radio of the WSN nodes. Duty cycling is based on the straightforward rationale that if the environmental parameters that a node is monitoring tend to stay constant over long periods of time and generating relatively light traffic compared to the full capacity, the node can conserve its energy resources by keeping its radio asleep much of the time and transmitting packets in a burst when the radio is active.

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 especially in the MAC protocol layer [1–5]. 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, for example, in the MAC protocols such as TMAC [6], AMAC [7], DSMAC [8], and CMAC [9].

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 possibly with high mobility. 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 network parameter adaptation strategies including radio duty cycle 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 intensive 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 network 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 capability. Our proposed approach, called PNAT (for *Predictive Network Adaptation by Tracking*) is a predictive framework for network adaptation 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 even beyond its direct sensing range.

Our framework for network adaption, PNAT, 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 from the bottom of the network protocol stack (i.e., MAC layer) 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 the MAC layer in anticipation of the arrival of the object in the near future, resulting in Predictive Duty Cycle Adaptation (PDCA) as an application of PNAT framework to the MAC layer.

A noteworthy aspect of the PDCA presented in this dissertation 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.

We then show how the predictive network adaptation by tracking (PNAT) framework can be further applied to a different layer at the same time, which turns out to be the application layer that controls the parameters for camera management. Conventional wireless sensor platforms [10] are equipped with minimal sensing and processing capabilities, and thus the energy dissipation by radio dominates the overall energy consumption. Wireless cameras, however, have much more complex hardware components and produce extremely high dimensional data, resulting in high energy consumption for data acquisition and processing. Since there are two major consumers of energy in WCNs: *radios* and *cameras*, therefore, it is also critical for overall energy efficiency that the camera also needs to be properly duty cycled as radio does if at all possible. In this dissertation, therefore, we present how the camera sensing rate can be dynamically adapted in a predictive manner using the PNAT framework, which will be called Predictive Sensing Rate Adaptation (PSRA). Note that a trivial approach that keys the duty cycling of the cameras off the duty cycling of the radios would not work in practice, however, since the relationship between the duty cycling of the radios and cameras must, at the least, depend on the state of the WCN. To elaborate, before a moving object can be tracked, it must be detected. When a node must engage in object detection, the off time for the camera cannot be allowed to exceed the time it would take for an object of interest to cross the field of view of the camera, while the radio is allowed to be at a very low duty cycle.

Below is a summary of the contributions made in this dissertation:

1. Developed an energy-efficient predictive network adaptation framework that can be applied to the multiple layers at the same time

2. Proposed the concept of the augmented sensing for object tracking in the network layers

3. Demonstrated the first use of the single-bit-long summary of the cross-layer information in the MAC header for predictive duty cycle adaptation without any communication overhead

4. Developed an efficient algorithm for avoiding synchronization failures when adapting the radio duty cycle in synchronous MAC protocols

5. Validated the proposed framework applied in large-scale simulations and in a real WCN testbed that consists of 13 iMote2-based wireless cameras with a realistic cluster-based distributed object tracking application

In the rest of the dissertation, we first review in Chapter 2 unique features of and challenges in wireless camera networks. Chapter 3 then provides a review of the relevant literature on adaptation approaches on network parameters in a single layer or multiple layers. The new PNAT framework is presented in Chapter 4 and 5 and its application to duty cycle adaptation of radio and camera in Chapter 6. The performance evaluation of the framework is presented in Chapter 7. As already mentioned, our performance evaluation is based on large-scale simulations and on a real Imote2-based testbed. Chapter 8 then concludes the paper.

2. CHALLENGES IN WIRELESS CAMERA NETWORKS

As a sub-category of the more general WSNs, the wireless camera networks (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 [11, 12]. It is obvious that the occurrence of an event/object of interest within the network makes the network more dynamic in terms of the state of the camera nodes and the traffic pattern.

In addition to their being event-driven, WCNs differ from the typical WSNs also from the fact the cameras are directional sensors due to its unique sensing model. A camera is typically modeled as a pinhole camera with limited visibility bounded by a fixed range of depth and viewing angle as shown in Figure 2.1. So it is possible for two or more nodes to see a target simultaneously even when the nodes are well separated by other intervening nodes that cannot see the target at all.

In this chapter, we focus on three unique characteristics of WCNs that make the system design extremely challenging. For a more comprehensive survey on camera sensor networks, the reader is referred to [14] and [15].

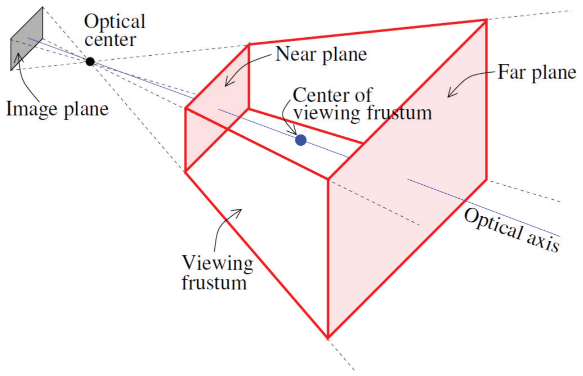


Figure 2.1.: Viewing frustum of a pinhole camera (reproduced from [13])

2.1 Traffic Patterns for Event-driven Collaborative Processing

WCNs differ from the more traditional WSNs in that the events occurring in the environment usually cause spatially-correlated traffic from multiple sources in the vicinity of the events. Since the traffic is generated among the nodes around an event in a bursty fashion during the presence of the event, it tends to be *temporal and locally intensive*. Therefore, the network-wide variation of the traffic patterns is much larger than that in conventional WSNs where the traffic generation tends to be periodic and distributed over the entire network and remains unchanged in the course of time.

Collaborative processing of visual data for better understanding of the environment even more enhances this locally intensive temporal traffic. Due to limited computational power and sensing capability, the sensor nodes in a WCN usually collaborate with one another in order to detect events of interest and to estimate their various attributes in a collective way. This 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. For WCNs, on the other hand, the nodes may be called upon to not only detect the presence of humans/objects in the environment but to also follow the movement of the detected humans/objects while exchanging their local measurements to each other.

For tasks such as object detection and tracking, a WCN may involve computations beyond the capabilities of the processor at any single node. Such tasks would require cluster-based distributed implementations of the algorithms as in [12, 16]. The nodes in a WCN may have to collaborate to estimate the various attributes of the objects of interest in order to surmount the extremely limited computational power available at the individual nodes. The collaborative processing that WCN nodes engage in is carried out with the help of clusters. That is, the nodes are allowed to form clusters with the expectation that it is the cluster as a whole that would “understand” an object in the environment. Clusters will usually elect cluster leaders in order to reduce the communication requirements when the network is either fielding a human query or when a cluster is communicating with another cluster.

The following computations are typical of this cluster based approach to collaborative sensing by the nodes of a network [16]: 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. Focusing on the fourth category listed above, consider for example the case where a cluster is trying to estimate a color histogram for an object that is visible to all the members of the cluster, our goal being for the sensor network to track the object. The cluster leader may assign the different bins of the histogram to the different members in the cluster and request that each member transmit the bin counts back to the leader. As each cluster member finishes its assigned task, all of the members trying to reach the cluster leader at approximately the same time with their bin counts would result in a burst of communication activity, with attendant packet collisions and wasted energy. The communication pattern among the cluster members would probably become even more vulnerable to effects such as the hidden terminal problem if the members collaborate in a distributed execution of a more sophisticated computer vision algorithm (as in the distributed implementation of, say, a Kalman filter).

For further detailed discussion, consider how a laboratory-based WCN could track simple objects moving about in its environment. Regardless of the specifics of the vision algorithms used, Figure 2.2 is good depiction of how a cluster of nodes working cooperatively would go about first detecting and then confirming the presence of an object in the portion of that space that all the cameras in the cluster can see. We can consider Figure 2.2 to be a general state transition diagram that could be instantiated for any specific vision algorithm. To drive home the point about the usefulness of this state transition diagram, let's briefly consider how the diagram would work for the same example that a histogram based approach is used for object

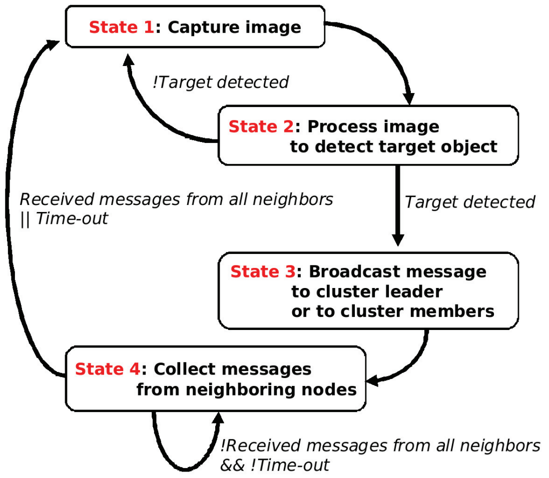

State 1: Capture image
!Target detected
State 2: Process image to detect target object
Target detected
Received messages from all neighbors || Time-out
State 3: Broadcast message to cluster leader or to cluster members
State 4: Collect messages from neighboring nodes
!Received messages from all neighbors && !Time-out

Figure 2.2.: Sensory processing state transition diagram at a camera node for tracking objects (reproduced from [17])

detection, recognition, and localization. In the state transition diagram shown above, for this specific collaborative vision process, State 1 corresponds to capturing the image periodically at each member node and State 2 to applying a threshold to the image at each member node. State 2 would also consist of accumulating the counts in the bin assigned to the cluster member; if the bin counts are below a threshold, the node assumes that there does not exist anything of statistical significance to report for that bin. State 3 would consist of reporting the results accumulated to the cluster leader. States 4 would apply only to the cluster leader; these states would enable the leader to collect the bin counts from the cluster members.

What is interesting is that the state transition diagram in Figure 2.2 is general enough to also represent a cluster head election process and to represent the idle state of a node if nothing statistically significant can be detected by the camera at the node. Let's first talk about the idle state. This is the state when the target cannot be discerned in the image recorded at a node. Obviously, in this state, the node will keep on capturing images and continue to stay in the idle state. Equally obviously, there will be no collaborative computing involving a node that is in the idle state. Regarding head election, initially all nodes that can discern an object features in their images would try to be cluster leaders. Every prospective cluster leader sends a message to the other members in its cluster about its leadership role. The actual leadership is acquired by the member who is the first at the inter-cluster communications. Mapping this process to the state transition diagram, State 3 corresponds to a member telling all other members that it has seen the object. State 4 in this case would entail each member relinquishing its leadership role to the member that was the first to broadcast its object detection. Obviously, these messages must be received within the timeout period shown in the diagram.

Recognizing that it would be impossible to create a truly application independent state transition diagram for the vision processes that one may wish to implement for collaborative computing in a WCN, we nonetheless wish to claim that the diagram of Figure 2.2 is of broad enough generality and that we may use it as a basis for creating a bursty communication model that would typify cluster-based processing of image data in such networks.

2.2 Resource Demands in Event-driven Collaborative Processing of Visual Data

In addition to the aforementioned communication activities (which corresponds to Step 3 and 4 in the state transition diagram in Figure 2.2) that pushes the radio bandwidth usage and energy consumption by the radio to the edge, all the image processing chain from image acquisition to low- to high-level processing such as feature extraction, object detection, tracking, and recognition (which corresponds to Step 1 and 2 in the state transition diagram in Figure 2.2) entails an extreme use of computing power and thus energy expenditure.

The cameras used in WCNs range from those that record low-resolution 160×120 grayscale images to those that record medium-resolution $640 \times 480 \times 3$ color images. Several low-level processing steps must be applied to the images for the extraction of information related to the identification of the object meant to be tracked. These steps may involve morphological operators, interest point extractors, edge extractors, image segmentation, etc. Given the constraints of real-time, it should be obvious that the peak demand on the resources required for processing the visual data in terms of the computing power and the memory could be very high and could involve high energy consumption.

To a certain extent, the resource requirements with regard to the computational power can be somewhat mitigated by having all the nodes that see a given object to engage in collaborative processing of the images of that object, which is described in the previous section yet at the cost of communication and energy overhead.

2.3 Quality of Service Issues

The Quality-of-Service (QoS) criteria for WCNs must of necessity be more stringent as compared to those for the more conventional WSNs. Due to the real-time nature of the WCN applications, any latencies and packet losses caused by, for example, duty cycling and 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. Since objects must be tracked in real time, the measurements made at a node have only a limited lifetime. That is, unless a measurement is reported to the cluster head that is in charge of estimating the target motion attributes from the measurements supplied by the cluster members within a specified timeout, it is a wasted measurement. When such latencies become excessive, a cluster may even loose track of the target. Low latency for data aggregation typically within a cluster, therefore, is a critical requirement to support the application-specific QoS in WCNs.

Achieving a high QoS by simply over-provisioning resources to 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. Increasing the duty cycle of radio or the sensing rate of the cameras would cause better application-level performance, but at the cost of high energy consumption.

Such demanding needs on the computational, communication, and energy resources shape the design space of the software used and the network protocols for wireless camera networks. Obviously, one must conserve the energy to the maximum extent possible. An important approach to energy conservation that has emerged over the years is network parameter adaptation to the locally prevailing conditions. Since the radios and the cameras are two large consumers of energy, it stands to reason that both should be subject to adaption to the locally prevailing conditions related to what is expected of the network.

Most of the adaptation strategies that have been proposed in the past, however, address network parameter adaptation only for a single layer or only in a reactive manner. That makes them unsuitable for the real-time constraints of a WCN. Note that in conventional WSNs with light-weight applications for periodic monitoring of the environment with scalar sensors, the energy consumption is dominated by the radios. The work that has been reported on single-layer adaptation [6–8, 18–24] has therefore focused only on the radios.

Reactive network adaptation approaches also have limitations on conserving energy while supporting application-level performance. The reactive approaches in general [24–29] and more specifically in the MAC layer [1–9,21–23] must entail an inherent delay in between the time of occurrence of an event as detected by a node and the time when the network adaptation takes place at the node. This delay can degrade the application-level performance significantly especially in real-time object tracking applications in terms of tracking accuracy and clustering operations. In fact, most existing MAC protocols, although possessing adaptation capability, have been designed with little consideration for being responsive in real time [30]. The extent of energy reduction achieved with such strategies is also limited since a camera node with a reactive approach cannot make predictions on the state of an object beyond its sensing range, resulting in its camera to be highly activated all the time.

The more detailed survey of the literature will be presented in the following chapter.

3. LITERATURE SURVEY

In this chapter, we will start our discussion by reviewing the existing literature on MAC protocols and related concepts to our proposed approach. Since our framework heavily relies on the interaction of the MAC layer with others and primarily applied to the MAC layer, the first part of our survey will focus on approaches in existing MAC protocols. We then review adaptation strategies and tracking methods in a broader context since our framework spans not only a single layer but multiple layers including MAC and application layers.

Keeping in mind that there is no single communication protocol that is suitable for all application domains in WSNs, we will then classify MAC protocols according to the best suited application space for the target protocols. The best suited application space for a particular class of protocols is identified based on qualitative analysis on their design itself which may or may not support the traffic patterns of their application in mind. We will see that many of the existing protocols do not successfully achieve their goals due to lack of careful investigation on the communication characteristics and requirements of their applications. It is especially true for event-driven MAC protocols, the application characteristics/requirements of which is analyzed in the previous chapter. Beside the qualitative analysis, the investigation on to what extent the communication protocols actually achieve their goals is beyond the scope of this survey since it requires quantitative analysis based on carefully designed and thorough experimental evaluations.

To design a MAC protocol for an application that may produce highly varying traffic loads possibly only within a local area, on the other hand, it is inevitable to employ an adaptive mechanism that adjusts the network parameters of a subset of nodes involved in any certain activity of interest. Depending on how to adapt the parameters based on what criteria, therefore, the applicability of an adaptive mechanism of a particular protocol shapes its application space that it can support. Our survey will be taken taking these into considerations in classifying the MAC protocols.

Among various MAC protocols, we will limit the scope of our survey within *duty cycling* and *random access* MAC protocols using a *single channel* with a *single radio*, since others are not directly relevant to our discussion. A more general survey on MAC protocols can be found in [31, 32].

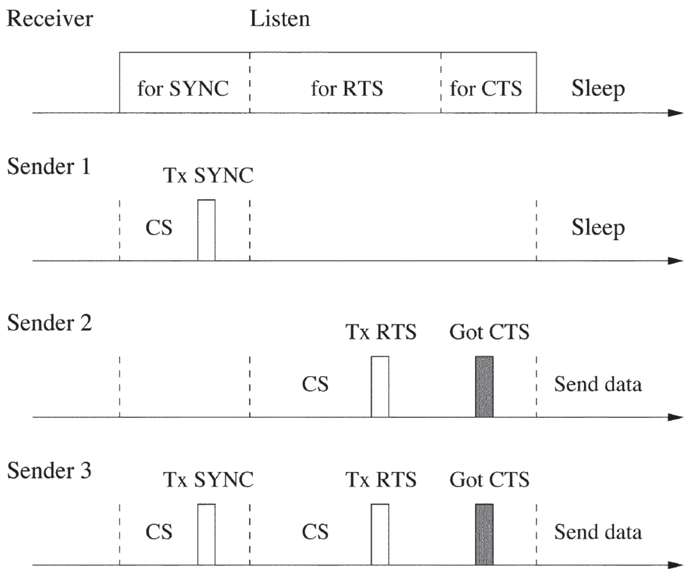
Afterward, we will continue to survey several concepts related to our approach in other domains such as predictive models in network adaptation and tracking methods in sensor networks especially using binary sensors. The event tracking methodology in our framework is based on detecting a single bit embedded in the packets and thus can be viewed as a type of object tracking in binary sensor networks. By reviewing existing binary sensor-based tracking methodologies, we will qualitatively analyze how the network parameter adaptation can be carried out with help of statistical estimation of the network state in terms of objects.

3.1 Approaches of MAC Protocols for Wireless Sensor Networks

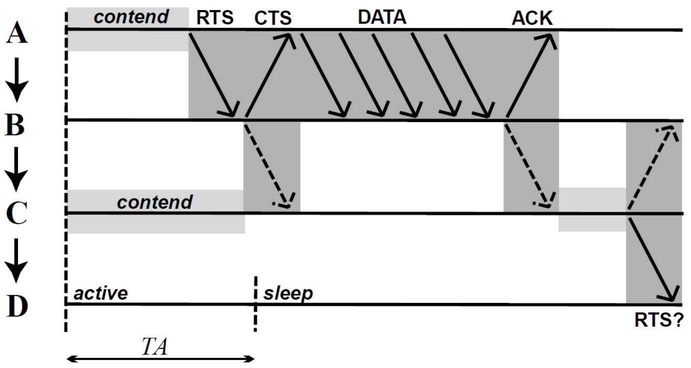
Among various MAC techniques, duty cycling-based on Carrier Sense Multiple Access (CSMA) [33] is one of the main approaches to achieve a low-power operation of nodes in wireless sensor networks. By alternating sleep and active modes and by transmitting data only during the active mode, a node can transmit packets in a burst only in a fraction of time and avoid unnecessary energy consumption and therefore prolong its lifetime. Generally, proposed MAC protocols that adopt this duty cycling technique can be bifurcated into *synchronous* and *asynchronous* approaches.

3.1.1 Synchronous MAC Protocols

The synchronous approaches use synchronization to assure that all or a subset of nodes in a network operates concurrently in such a way that they are planning their



(a)



(b)

Figure 3.1.: Timelines of SMAC and TMAC: (a) Timing relationship between senders and receiver in SMAC (b) The early sleeping problem of TMAC (reproduced from [5, 6], respectively)

communication activities according to their neighbors' schedule, so that they share a period of time to communicate. A node running SMAC (Sensor MAC) [5] periodically sleeps and wakes up simultaneously with its neighbors. SMAC divides the time frame into active and sleep period, and active period is further divided into SYNC, RTS (ready to send), and CTS (clear to send), as shown in Figure 3.1 (a). The length of the sleep period determines the duty cycle of SMAC, since that of the active period is fixed. During SYNC period, nodes need to exchange synchronization information periodically to reset the clock drift among neighbors. If there is data to transfer, then the sender and receiver establish a connection using RTS-CTS handshake to avoid collisions with other neighbors. Then the data transfer takes place while the other nodes return to sleep. This RTS-CTS handshaking is employed to alleviate so-called Hidden Terminal Problem among two-hop neighborhoods that causes packet collisions. A node only listens during the contention periods for SYNC and RTS briefly unless it has data to send. Only nodes participating in data exchange stay active after these contention periods, whereas others can return to sleep. SMAC also employs *adaptive listening* to enable immediate message passing right after the end of previous communication in order to reduce per-hop latency. Since SMAC may create multiple virtual clusters in a network that have the different schedules, there are the nodes belonging to multiple virtual clusters at the border of the virtual clusters, called the border nodes. Since these nodes should wake up according to each of the schedules of the virtual clusters, they dissipate energy faster than the non-border nodes due to frequent idle listening and overhearing. In addition, the predetermined and fixed lengths of sleep and active periods decrease the performance of the protocol under variable traffic loads.

TMAC (Time-out MAC) [6] improves the energy efficiency of SMAC by *adaptively* reducing energy waste in active period based on the observation that if a node has data to send, then the transmission will take place at the beginning of RTS period with, of course, a short random backoff. TMAC uses a short window, that is *time-out,* at the beginning of RTS period to determine whether it remains listening further

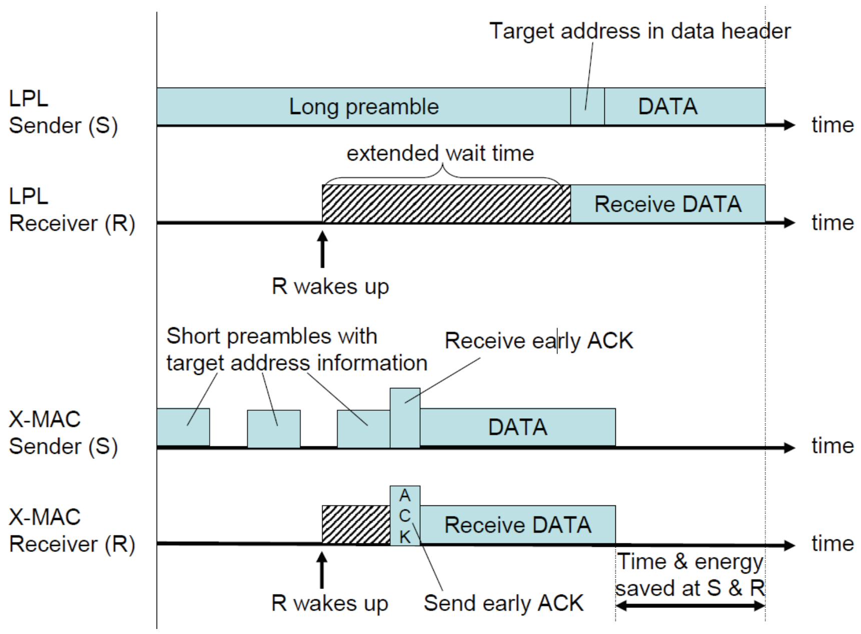


Figure 3.2.: Timelines of the extended preamble of Low Power Listening (LPL) and the short preamble approach of X-MAC (reproduced from [2])

or not. When there is no activity in this short window, it allows a node to sleep earlier, saving energy. Yet it also causes a node to sleep too early even if its neighbor may have packets to transmit in the middle of the contention period, called early sleeping problem as shown in Figure 3.1 (b). Thus, nodes may suffer from poor throughput in case of heavy traffic loads. To mitigate this problem, TMAC employs FRTS (Future Request-To-Send) that lets two-hop neighbors not to sleep early. But, it only decreases the effect of the early sleeping problem from one-hop range to two-hop range. Thus, the problem still remains. Rather, it may enhance the contention in the region of current traffic due to increased communication overhead, resulting in poor throughput at the center of active region.

A comparison of duty cycling MAC protocols for WSNs can be found in [34].

3.1.2 Asynchronous MAC Protocols

The asynchronous MAC protocols, on the other hand, employ an extended preamble approach in packet transmission and low power listening or preamble sampling in packet reception with asynchronous schedules. B-MAC (Berkeley MAC) [3] senses a channel to avoid collision or to check any ongoing communication activity by sampling signal strength and detecting outliers, called *clear channel assessment* (CCA). The rationale behind it is that in case of signal due to extended preamble or data, the samples of the signal strength would be consistent whereas in case of no signal, it would not due to white noise, resulting in frequent outlier occurrences. Whenever a node wakes up, it samples the channel multiple times with a certain sampling interval using CCA, called *low power listening* (LPL). Thus, the preamble should be long enough to be checked by other nodes for reliable communication since they do not know the schedules of their neighbors. If the channel checking interval (i.e., wake-up interval) is shorter, implying that potential receivers check the channel more frequently, then the potential receivers will spend more energy. Yet at the cost of receivers' energy, the potential senders can send a shorter preamble, resulting in low latency. Longer preambles, on the other hand, enables a longer sleep interval between channel sensings, which helps energy conservation of potential receivers. When a node is not the intended recipient of the current communication activity, however, long preamble causes long idle listening of the non-intended receivers, ending up with unnecessary energy waste of the nodes. Therefore, there is a trade-off between the length of a preamble in sender-side and the channel checking interval (i.e., wake-up interval), determining to which side the communication burden is given more. Since B-MAC pursues low complexity and small size, it instead provides configurable interfaces that allow upper-layer services to adjust its operation mode according to runtime network conditions.

WiseMAC (Wireless Sensor MAC) [4] uses a preamble sampling similar to LPL in B-MAC. In contrast to B-MAC, a transmitter can *learn* its receiver's schedule in

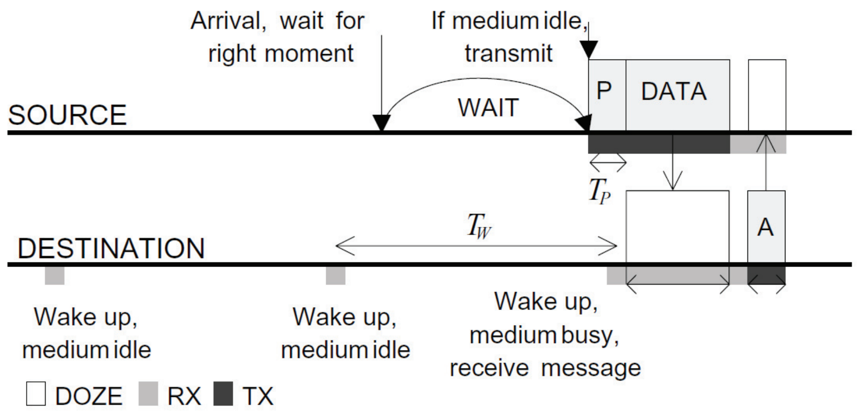


Figure 3.3.: Timelines of a pair of source and destination in WiseMAC (reproduced from [4])

ACK packets from the receiver by piggybacking, so that the transmitter can initiate its preamble only a short period of time before the receiver's active period, as shown in Figure 3.3. The preamble transmission duration is determined with taking into account the clock drift uncertainty from the time when the last packet exchange took place between the sender-receiver pair. This reduces the length of the extended preamble, an energy problem in B-MAC.

While B-MAC and WiseMAC provide simple yet efficient schemes for low power communication in an asynchronous way, they are inherently expensive to perform broadcasting packets to neighbors and have no mechanism for the hidden-terminal problem. In addition, per-hop delay introduced by extended preamble can be accumulated at multi-hop communication, degrading latency performance. Thus, these protocols mainly target delay-tolerant applications, such as environmental monitoring, that do not require real-time operation. To resolve the latency issue, an approach [35] is proposed that attempts to minimize the per-hop latency in WiseMAC by multi-hop cross-layer design. For the broadcasting issue, another approach [36] is proposed that reduces the cost of broadcasting in WiseMAC using the k-Best-Instants approach that calculates a minimum set of instants that can cover asynchronous wake-ups of all neighboring nodes based on their schedules.

In addition to these inherent barriers in asynchronous MAC protocols, LPL-based approaches encounter difficulties when they are applied to 802.15.4 radios since LPL is based on bit-wise operation while 802.15.4 is packet-based, resulting in solutions like X-MAC (Short Preamble MAC) [2]. Instead of continuous bit stream-based preamble, X-MAC sends a chain of consecutive short preamble packets, each of which contains the address of an intended receiver. Then, if the intended receiver wakes up and listens these short preamble packets, then it will send an *early ACK* between the short preamble packets, stopping further excessive preamble transmission of sender, while the other non-intended receivers can return to sleep early by overhearing a short preamble packet, as shown in Figure 3.2. The early sleep of non-intended receivers reduces energy spent by unnecessary overhearing while the early ACK reduces per-

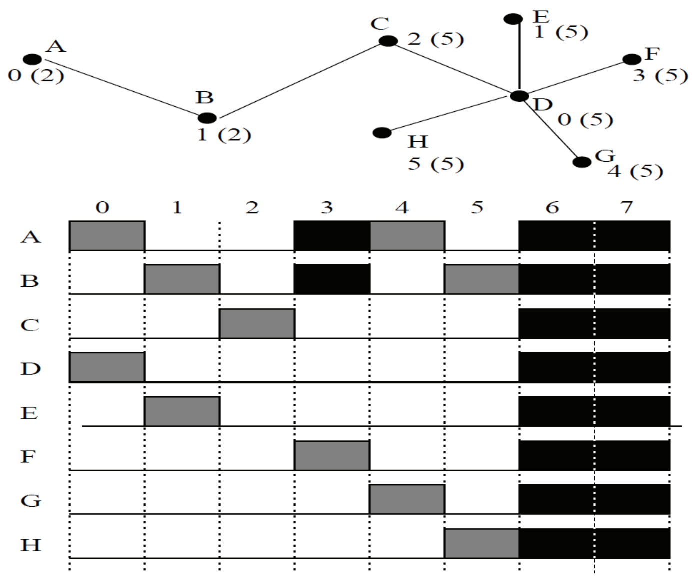


Figure 3.4.: The time frame rule of Z-MAC: The numbers assigned in the topology indicate the slot numbers assigned by DRAND which correspond to the shaded slot in the bottom, and the numbers in the parenthesis are the maximum slot number within two-hop neighbors. The dark slots are the empty slots that can be used by any node with random access. (reproduced from [1])

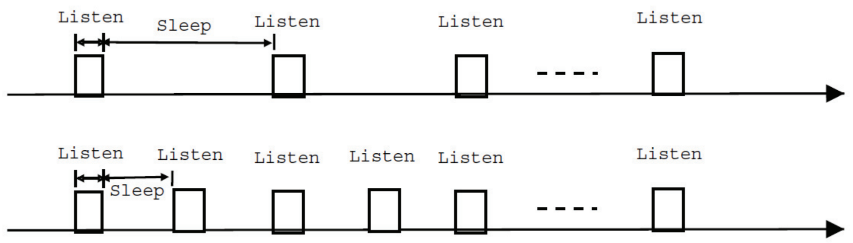
hop latency as well as energy spent by both the sender and the intended receiver. Based on the observation that fixed schedule MAC protocols are always sub-optimal under time-varying network conditions, X-MAC also *adapts* to variable traffic load by dynamically tuning the durations of receiver's sleep and listen periods. As the measure of traffic load, the probability of receiving a packet within a timebound is estimated by observing the packet arrival rate at a node. Besides of these efforts, however, it also has limitations; The usage of the early ACK is limited only to unicast message exchanges, and X-MAC does not provide a mechanism to avoid the hidden terminal problem.

3.1.3 Hybrid MAC Protocols

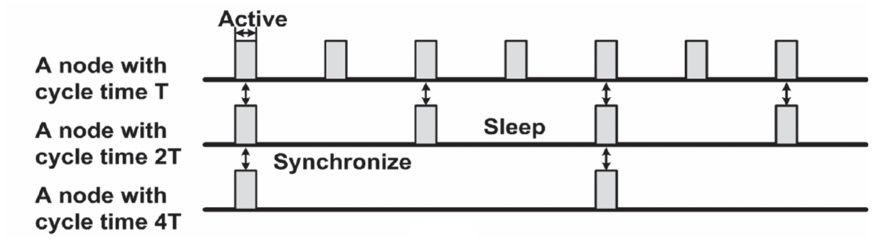
While a pure TDMA scheme has been considered an impractical solution for WSNs due to its poor scalability, poor adaptability to changing network conditions, and inefficient broadcasting capabilities, as already indicated in other sources [1], there has been several efforts to combine the strengths of CSMA- and TDMA-based schemes, leading to a *hybrid* approach. In fact, SMAC and TMAC are hybrids of CSMA and TDMA in that they divide an active period into several specific time slots such as SYNC, RTS, CTS, and DATA while the underlying medium access method for each slot is CSMA. In contrast, PTDMA [37] shows smooth switching between CSMA- and TDMA-based on the level of contention. Given a time slot assigned by a common TDMA scheme, the probability of accessing the slot by owners and non-owners is adjusted depending on contention.

Z-MAC (Zebra MAC) [1] improves PTDMA in such a way that there may be more than one owners per slot if the owners are apart beyond two-hop neighborhoods, which is similar to resource allocation in cellular networks. An owner of a slot is determined only once at the very beginning stage, and it has earlier chances to get the slot due to a higher priority over non-owners based on pre-determined contention window sizes. The transmission control scheme of Z-MAC lets any node can compete for a slot yet with different probabilities in low contention while the owner of the slot and its one-hop neighbors are allowed to compete in high contention. An example of schedules of nodes is presented in Figure 3.4 where although the global time frame size is six, the local time frame size of Node A and B is four due to the time frame rule in Z-MAC that allows re-utilization of slots among nodes beyond two-hop range.

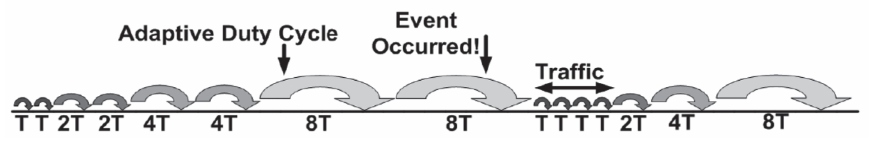
Z-MAC is robust to the hidden terminal problem since when time slots are assigned to nodes, DRAND (Distributed Randomized TDMA Scheduling) [38] prevents two nodes within a two-hop communication range from being assigned to the same slot. To further alleviate the hidden terminal problem, Z-MAC employs explicit contention notification (ECN) in high contention. ECN is similar to RTS yet differs by sending only in high contention only to two-hop neighbors of the current slot. When a node A experiences high contention to another node B, then the node A sends one-hop ECN to the node B, triggering the node B to broadcast two-hop ECN.



(a)



(b)



(c)

Figure 3.5.: Nodes with different duty cycles in (a) DSMAC and (b) AMAC, respectively, and dynamic cycle time adjustment of (c) AMAC (reproduced from [7, 8])

3.1.4 Adaptive Synchronous MAC Protocols

In addition to these synchronous and asynchronous approaches, further improvements on the tradeoff between energy efficiency and latency can be achieved by more explicit *adaptive* scheduling of sensor nodes. Note that many of the aforementioned protocols such as SMAC and TMAC also have an adaptive mechanism in terms of duty cycle. And, LPL-based approaches including BMAC are inherently adaptive in terms of duty cycle when packets are consecutive. Yet the criteria for adaptation is rather simple, for example, the presence of any subsequent packets. Thus, although they improve per-hop latency to some extent, the effect of the adaptation is relatively limited. In this section and the following, we will consider MAC protocols that explicitly claim an adaptive method that employ more sophisticated adaptation criteria or mechanisms.

DSMAC (Dynamic SMAC) [8] is a variant of SMAC and dynamically adjusts duty cycle according to the current traffic condition and energy utilization efficiency. The current traffic condition is measured based on *average one-hop latency*, which is then embedded in SYNC packet by a sender node. This information is retrieved and used together with energy utilization efficiency by receiver nodes to *exponentially* adjust their duty cycle as shown in Figure 3.5 (a). This exponential duty cycle change is achieved by adding extra wake-ups and thus still provides shared time intervals for communication among heterogeneous schedules. When DSMAC experiences heavy traffic, meaning that average latency is larger than a threshold, then DSMAC doubles its duty cycle to provide larger throughput. If the average latency decreases, then it halves. The adjusted new schedule is put into the next sending SYNC packet to inform neighboring nodes of the schedule change and thus to be utilized. Since DSMAC adjusts is parameters based on the average network condition, it takes time to perform the adaptation; in other words, it has lag in adaptation. The lagging time strongly depends on how to estimate the average, for example, the size of the window to compute the average.

AMAC (Adaptive MAC) [7] has a similar approach to adjusting the duty cycle as that of DSMAC like Figure 3.5 (b), yet differs by putting all the nodes that receive *any* traffic to the *maximum* duty cycle as shown in Figure 3.5 (c). In other words, AMAC doubles and halves its cycle time adaptively like DSMAC; however, it reduces the cycle time of a node to the minimum (i.e., increasing duty cycle to the maximum) when the node receives a RTS packet while it doubles the cycle time when there is no traffic for a period of time until its cycle time reaches to the maximum (i.e., minimum duty cycle). The motivation that drives this design is that all the nodes along a routing path should be on a high duty cycle to maximize throughput and minimize latency while other nodes conserve energy in low duty cycle mode. Also, AMAC removes unnecessary RTS periods as in SMAC using the modified SYNC packet that contains a preamble bit, called the communication SYNC, to avoid idle listening of nodes during RTS period. Thus, when a node has a data to send, it broadcasts a communication SYNC serving as pre-RTS during SYNC period. If nodes hear the communication SYNC, then they extend their active period to listen for RTS to check whether they are intended recipient or not. Otherwise, they go to sleep right after the SYNC period. Using communication SYNC packets, however, increases the probability of packet collisions in SYNC period, resulting in better chance of synchronization failure among nodes. This is especially true for the case of collaborative processing of event-triggered data. It could be worse if the event has high mobility and continuously moves and thus keeps triggering schedule changes of nodes around the event.

While DSMAC and AMAC allow nodes to have different schedules among neighbors, they do not provide any mechanism that enables efficient communication between neighbors with different schedules *only within* the shared time intervals between them. Without this mechanism, a node may attempt to transmit a packet to its neighbor that has a different schedule when the neighbor is not on duty in this period. Then, the node may retry the transmission multiple times, wasting unnecessary energy. In Section 6.1.4, we propose a detailed algorithm that enables successful communication among heterogeneous schedules.

3.1.5 Adaptive Asynchronous MAC Protocols

While DSMAC and AMAC can be thought of as synchronous MAC protocols with exponential duty cycle adaptation scheme, there are several adaptive MAC protocols based on asynchronous schedules [21–23]. MaxMAC (Maximally Traffic-adaptive MAC) [21] is an adaptive asynchronous MAC protocol that adapts the duty cycle of nodes *exponentially* on top of WiseMAC. Besides the short preamble nature of WiseMAC, MaxMAC adds online traffic adaptation mechanism that puts extra wake-ups between regular wake-ups of the base schedule, as shown in Figure 3.6.

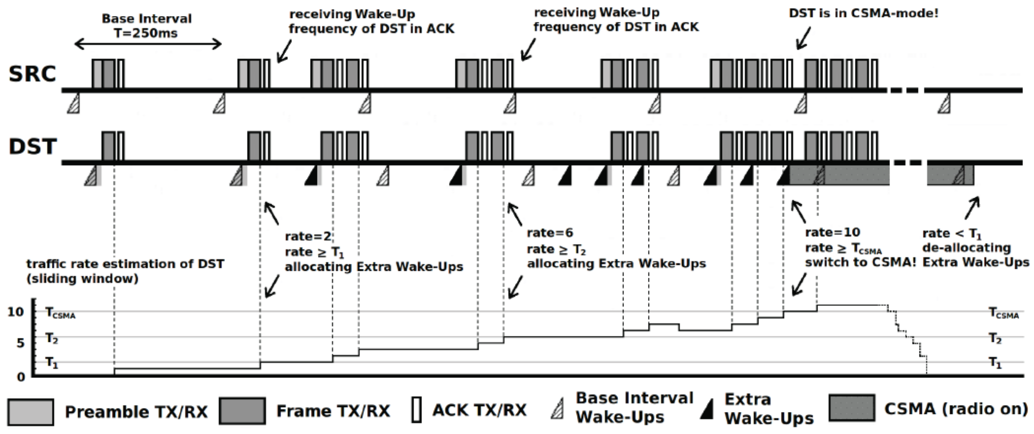
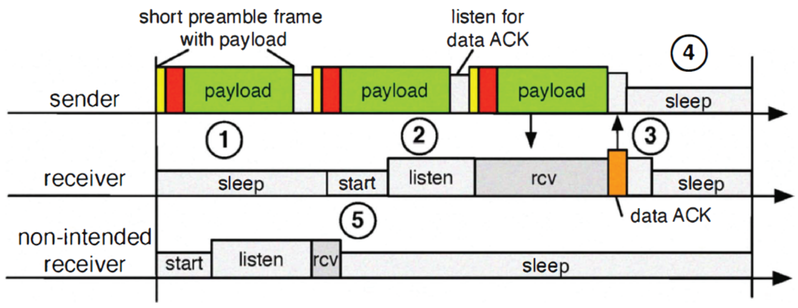


Figure 3.6.: Adding extra wake-ups in MaxMAC according to varying incoming traffic (reproduced from [22])

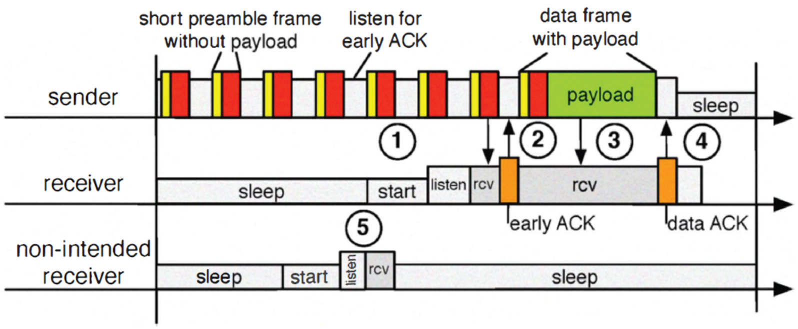
This adaptation takes place based on *the current rate of incoming traffic* estimated using a sliding window of one second. The schedule change is achieved among three steps with predefined traffic thresholds through a soft-state approach with predefined timespans, resulting in avoidance of frequent state transitions.

BEAM (Burst-Aware Energy-Efficient Adaptive MAC) [22] is also an asynchronous adaptive MAC protocol that aims to resolve drawbacks of X-MAC (1) by adding acknowledgment-based link-layer reliability support on top of X-MAC and (2) by adding receiver-initiated transmission rate control of senders and sender-initiated duty cycle adaptation of receivers using traffic and buffer indicators in the MAC header. The main drawback of X-MAC is pointed out that it works well only with light traffic condition. Under heavy traffic or high link error rates, it suffers from significantly reduced throughput due to lack of reliability support. BEAM, on the other hand, employs both early ACK for successful preamble reception and data ACK for successful data reception to provide reliable communication. It basically utilizes two types of short preambles *with* and *without* payload, as shown in Figure 3.7 (a) and (b), respectively. The short preamble approach without payload which is similar to X-MAC is more energy efficient in sending preambles yet less robust and more complex compared to the basic operation mode that sends the short preambles with payload. Thus, BEAM alternates between two modes depending on the size of the payload: for smaller payload size, the approach in Figure 3.7 (a) is preferred and vice versa.

Another drawback of X-MAC that BEAM tackles is lack of adaptation scheme to the highly varying traffic loads. When a *sender* experiences congestion based on its *current buffer state*, then it sets one bit-long *traffic indicator* defined in the FCF field of 802.15.4 MAC header of outgoing packets for short preamble or data, so that the receiver adapts its wake-up interval (i.e., duty cycle) by estimating an earlier time to wake-up according to the traffic indicator. When a *receiver* suffers from congestion due to possible buffer overflow, on the other hand, it sets two bit-long *buffer indicator* in the same FCF field of outgoing packets for early ACK or DATA



(a)



(b)

Figure 3.7.: Short preamble approach of BEAM (a) with and (b) without payload (reproduced from [22])

ACK, causing the sender to reduce its transmission rate. The former sender-initiated congestion control is called Listen Cycle Adaptation, while the latter receiver-initiated one is Transmission Cycle Adaptation. According to these adaptation schemes, the authors claim that BEAM can react and adapt to the rapid traffic change so that it can handle both low traffic and local/bursty event-driven traffic.

While MaxMAC and BEAM are built on top of WiseMAC and X-MAC, respectively, ALPL (Adaptive LPL) [23] is based on B-MAC and adapts the listening (operation) mode of B-MAC on-the-fly while it also adapts the routing path with a cost function based on the state information of one-hop neighbors, resulting in a cross-layer optimization in a greedy sense. In each periodic routing update messages in a proactive tree-based routing protocol employed, state information is embedded such as the current listening mode, the number of descendants, the current duty cycle, and its role. Since the listening mode of B-MAC determines the listen check interval of the receiver, it also determines the preamble length of the potential sender, correspondingly. Given these state information of neighbors, a node adapts its listening mode in a MAC layer based on the *incoming traffic rate*. If incoming traffic rate is not uniform over the network, then the duty cycle of a node will be different from its neighbors, resulting in non-uniform cost and thus different routing decisions in each node. Also, If a node detects an event, it may alter sensing activity level, implying it has a temporal role different from its neighbors, yielding different cost. These factors cause routing path to circumvent the busy nodes.

3.1.6 Event-driven MAC Protocols

While most of the aforementioned MAC protocols are designed for a wide range of applications in general wireless sensor networks, several protocols (e.g., Sift [39], CC-MAC [40], Alert [41], and EC-MAC [42]) are explicitly and specifically designed for *event-driven* wireless sensor networks where the event-triggered packets should be handled typically with low latency. Since the timeliness is critical when an event of

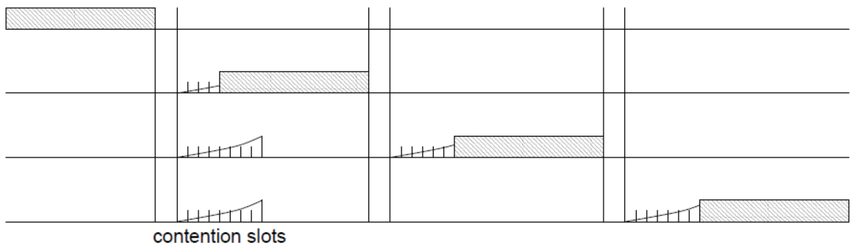


Figure 3.8.: A timeline of four nodes running the Sift protocol. Shaded bars indicate the duration of packet transmissions (reproduced from [39])

interest occurs, the energy conservation is set to be a secondary goal when the event is present.

Sift [39] adapts $CSMA/p^*$in such a way that each node has non-uniform probability distribution over contention windows for transmission at each time slot—each time slot is divided into multiple contention windows, as shown in Figure 3.8 (a). Sift multiplicatively increases the probability in the next contention window if no node transmits in the current contention window. This enables very low contention at the beginning of each time slot so that first few messages can be transmitted in a timely fashion. This approach is based on the assumption that only a subset of reports on events of interest is enough to be transmitted as long as it *is* reported to the data sink. It is because the underlying rationale is the packets generated in the vicinity of an event are spatially-correlated thus redundant.

CC-MAC (spatial Correlation-based Collaborative MAC) [40] also exploits the redundancy in generated packets to minimize the number of packet transmissions and thus packets collisions and energy waste, correspondingly. Since CC-MAC has the same rationale as Sift, it instead selects a subset of nodes iteratively in such a way that the nodes have maximum coverage with minimum overlapped sensing area since measurements from overlapped area is redundant. The measurements from the selected nodes are merged together in the middle of the routing path to construct a big packet for efficient routing.

These protocols are all based on the belief that multiple measurements to the same event must be redundant and thus many of them can be discarded. This may be suitable for simple scalar sensor-based networks that do not require local collaborative and collective reasoning around events of interest; however, it does not hold for a class of sensor networks that employ sophisticated sensors, such as cameras. In such networks, each measurement taken from different sensors about the same event have its own perspective and meaning. Therefore, although there exists some redundancy, all of them need to be processed if possible.

Alert [41] tackles the similar problem of Sift and CC-MAC yet with different method and different philosophy. The goal of Alert is how to minimize the latency of all event-triggered packets in event-driven WSNs although there will be some additional energy consumption.. Alert employs a combination of time and frequency multiplexing with multiple channels. Due to multiple channels, the contention is minimized by optimizing the channel selection probability of the nodes. Besides the methodological differences between Alert, Sift, and CC-MAC, an interesting observation among them is that the underlying assumption of Alert is exactly opposite to Sift and CC-MAC; Alert is designed with the belief that although the event-triggered messages are typically correlated, *all* messages as a whole—not just a *subset* of them as in Sift—provide valuable information for further inference or refinement of the decision about the events such as detection of false positives.

EC-MAC (Event-centric MAC) [42] is also proposed for event-driven data collection with asynchronous schedules. Yet the details will be discussed in Section 3.3.

3.2 Application Space of Adaptive MAC Protocols for Wireless Sensor Networks

While there exist many canonical MAC solutions that resolve energy problems in wireless sensor networks, there also have been proposed various types of adaptive methods that tackle variable traffic loads as described in the previous section.

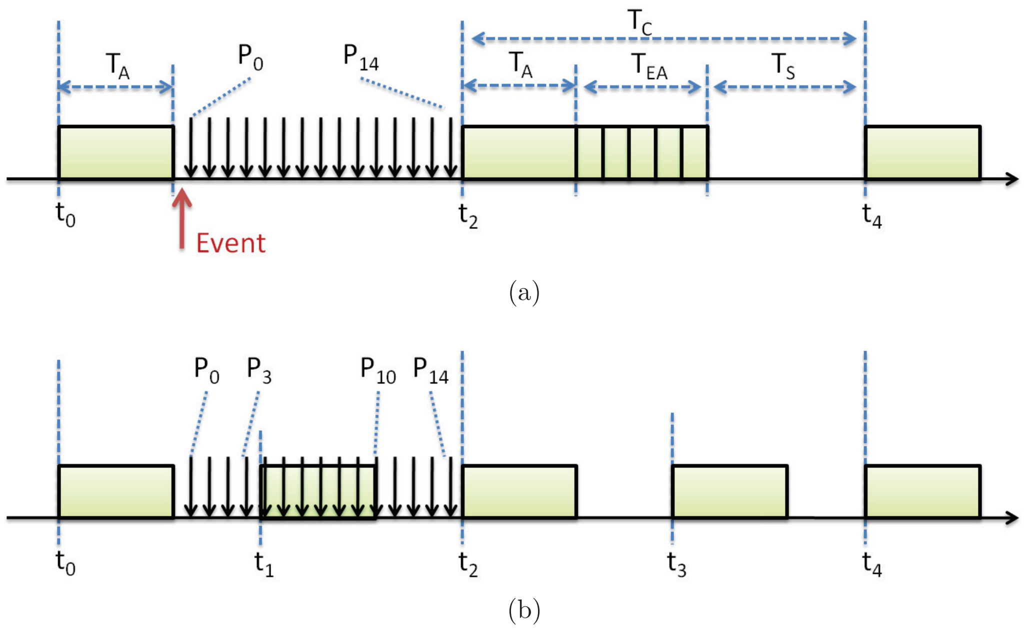


Figure 3.9.: An example of handling a chain of packets (from P_0 to P_{14}) generated by an event of interest during a sleep period with different adaptation methods of (a) Type A and (b) Type B. T_C denotes the period of a cycle consisting of an active and a sleep period, denoted as T_A and T_S, respectively. T_{EA} indicates the extended active period in (a).

Depending on how they adapt their network parameters based on what criteria, however, their adaptability and applicability greatly varies in the space of applications. In this section, therefore, we first classify the adaptive MAC protocols according to their adaptation criteria and methods, and then investigate their advantages and disadvantages to help finding the best application space.

3.2.1 Adaptation Method

In terms of the duty cycle adaptation, most adaptive MAC protocols employ either *dynamic active period with a fixed cycle* (Type A) or *dynamic cycle with a fixed active period* (Type B), where a cycle is defined as the duration of a pair of active and sleep periods. In other words, the former changes the duty cycle of a node by increasing/decreasing the duration of the active period while the latter by adding/removing additional wake-ups between two base wake-ups. MAC protocols of Type A include SMAC with Adaptive Listening, TMAC, and BMAC while those of Type B include DSMAC, AMAC, MaxMAC, and BEAM.

Type A approach is suitable for the case of a continuous stream of packets between nodes since active period will be prolonged as more packets are in the queue. Thus, for the packets that are *already* in the queue, they will be served with low latency. For the packets that are instantaneously generated in the middle of sleep period, however, they must stay in a queue before getting served with extended active period. In the case of low duty cycle mode of a Type A protocol, the sleep period is usually large, resulting in the introduction of a large latency in the packets generated during the sleep period, which would correspond to the packets at the beginning of a packet stream. Suppose an example that a chain of packets (from P_0 to P_{14}) are generated during a sleep period triggered by the detection of an event as shown in Figure 3.9 (a). In a worst case, the packet P_0 must stay in a queue for the entire sleep period to get served at the next active period beginning at time t_2. Yet all the packets will be served in the next cycle with extended active period, denoted as T_{EA} in the figure, if necessary.

Type B approach, on the other hand, can serve the packets generated in the middle of sleep period with less latency than Type A approach, since the duration of the sleep period is shorter as long as the nodes are in a high duty cycle mode. Suppose the same example that the same packet chain is generated yet with a high duty cycle as illustrated in Figure 3.9 (b). For the packet P_0 and P_{10}, it will be served with low latency at time t_1 and t_2, respectively, since the new active period begins. Provided that nodes are in a high duty cycle mode, therefore, Type B approach can handle better the potential packets that may be generated in the future sleep period while Type A approach does so for the already-generated packets. This implies that Type A approach is suitable for packet forwarding in the middle of a routing path while Type B approach is suitable for packet handling at the beginning of the routing path. Since Type B approach allows nodes to have different schedules with different duty cycles, however, it requires more complicated methods than Type A approach to maintain the heterogeneous schedules in a network and to efficiently communicate among them. Also, due to the increased number of wake-ups in a high duty cycle mode, energy is consumed more in Type B than in Type A.

3.2.2 Adaptation Criteria

Adaptations using one of the aforementioned approaches take place when a decision is made based on a certain type of traffic conditions that each protocol defines. Virtually, all of the adaptive MAC protocols that we have mentioned adapt their duty cycle based on the current traffic conditions. How to measure and estimate the current traffic, however, varies among the protocols.

The MAC protocols that adapts the duty cycle based on the *average* estimates of the parameters (Type C) such as the incoming packet rate (e.g., MaxMAC) or average one-hop latency (e.g., DSMAC) employ a relatively conservative congestion control mechanism compared to those based on the *instantaneous* conditions (Type D) such as the current buffer state (e.g.,TMAC, AMAC, and BEAM). Type C protocols inherently have lag in adaptation depending on to what extent it accommodates the history to estimate the average. Type D protocols, on the other hand, may frequently change the duty cycle or schedule while they are agile in adaptation. To avoid too frequent schedule changes, protocols may choose a soft-sate approach with a time-out as in MaxMAC.

Those protocols may change the duty cycle *gradually* (Type E) as the adaptation metric reaches one of the pre-defined thresholds (e.g., MaxMAC and DSMAC). This approach is a more conservative congestion control method than those who *immediately* and aggressively adapt the duty cycle (Type F) (e.g., TMAC and AMAC). Type F protocols may over-provide bandwidth and thus spend unnecessary energy; however, in case of a burst of upcoming packets triggered by an event of interest, they can be better prepared and thus handle them better with low latency.

The conservative approaches (Type C and E) tend to prioritize the energy efficiency over the performance even during the time period that adaptation is necessitated. Therefore, the conservative MAC approaches are more suitable for a class of applications that require tight energy efficiency and are tolerable to the network performance. Especially they are best suited to the applications that generate delay-tolerant data packets, since the conservative approaches may require a larger transient time to react to the change of network conditions and to adapt their parameters (e.g., the duty cycle) to the maximum. Of course, once the parameters are adapted to a desired level, then delay-sensitive data can be properly handled, and variable traffic loads can also be better handled by reducing buffer overflow problem. Due to the large transient time, however, when the source of the delay-sensitive data packets are not static and when the routing path of the delay-sensitive data packets in a multi-hop network is volatile or dynamic, then the performance may be significantly degraded since multiple transient times would be required for the network parameters to be adapted and stabilized.

The aggressive MAC approaches (Type D and F), on the other hand, can support better another class of applications that put more emphasis on the performance than energy efficiency during the time when the network needs to be highly activated. These type of MAC protocols are best suited to the applications that have a long period of idle time followed by a sporadic and short time of high activity in a network. It is because the MAC protocols can maximize the performance of applications yet may waste energy when they over-provide resources during the highly activated periods. These aggressive approaches minimize the transient time required to adapt the parameters so that delay-sensitive data can be treated better. Especially the combination of Type D and F (e.g., AMAC) can provide better support for even a light traffic of delay-sensitive data than the combination of Type C and E (e.g., DSMAC). A surveillance application in an intruder detection/tracking system could be an example of such applications.

3.2.3 Applicability to Collaborative Networks

In terms of the applicability of adaptive MAC protocols to the applications that may require collaborative processing among neighboring nodes, we need to investigate how well the MAC protocols cope with *broadcast* messages since such application require extensive broadcast-based traffic among mostly single-hop neighbors. This is especially true for tracking applications in wireless camera networks. For asynchronous MAC protocols, broadcasting is inherently an expensive operation when it is achieved by a series of unicast to each neighbor, or it induces a longer preamble to make sure all neighbors are listening, resulting in a larger per-message latency. For synchronous adaptive MAC protocols, maintaining and utilizing heterogeneous schedules among neighbors is not an easy task. Moreover, successfully transmitting a broadcast message among nodes that have heterogeneous schedules is even more difficult. To the best of my knowledge, there has been no synchronous MAC protocol that provides such functionalities.

Consider an object tracking application in a wireless camera network where objects have high mobility, which is prune to occur in reality. The MAC protocol employed in such network should be able to handle all generated traffic with minimum delay since time is critical in tracking. With highly mobile objects, the transient time of the MAC protocol is therefore also very critical in tracking performance. We can only try to minimize this transient time and cannot remove it as long as the MAC protocol *reacts* to the environment. And, the aforementioned adaptive MAC protocols all react to the *current* network conditions. Therefore, to the best of my knowledge, there is no way that removes this transient time in existing MAC protocols.

The only way that we can conceive is to adapt the network parameters *before* any communication activity occurs. Then any significant communication activity will be able to be served with optimal parameters, resulting in optimal performance. This type of *proactive*, not reactive, approaches require a measure to make decisions on when to and how to adapt the parameters. While the reactive approaches are based

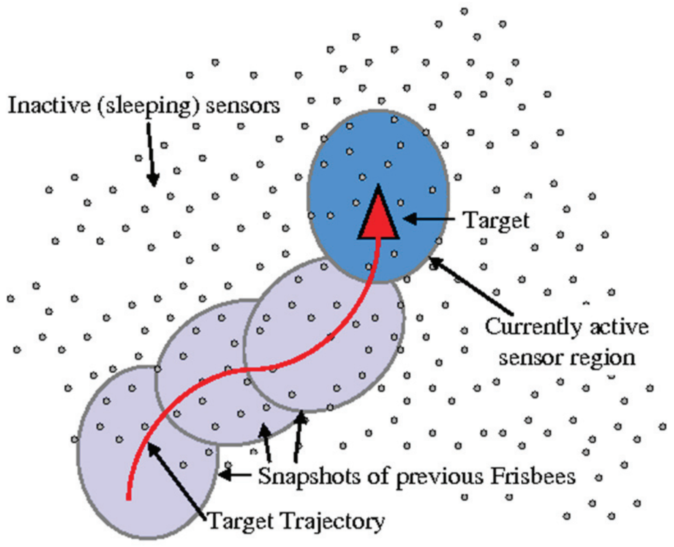

Inactive (sleeping) sensors
Target
Currently active sensor region
Snapshots of previous Frisbees
Target Trajectory

Figure 3.10.: The Frisbee model (reproduced from [25])

on the measure of the current states, the proactive approaches should be based on a certain type of *prediction* that gives us an estimation of the network conditions in the future.

3.3 More General Network Adaptation Strategies

There have been multiple approaches and models that somehow accommodate the concept of prediction in adapting a network. The approach in [25] is proposed with a belief that unattended networks deployed in a real environment must automatically self-configure to adapt to dynamic environmental conditions. As a core building block, the *Frisbee model* is proposed, which dynamically sets a circular zone with a pre-defined radius around the *current* location of a target being tracked by the network, as shown in Figure 3.10. The nodes within the zone are then kept in fully active state while those outside the zone are inactivated (i.e., power-saving mode) to minimize energy waste. As the target of interest moves, the zone should also move centered at the target. A node then makes a decision autonomously on whether it is within the range or not with the help of a localization algorithm.

To enable this heterogeneity, the Frisbee model assumes that it must be possible that the nodes in power-saving mode can be activated by other nodes using a wake-up signal. Therefore, as a target moves, a so-called “wake-up wavefront” is propagated in the direction of the movement of the target. Since inactivated nodes are not able to detect a target nor send any signal in this model, a set of nodes that stay awake all the time are defined and serve as “sentries”, resulting in a tiered architecture.

Since it uses a wake-up signal, however, the nodes must be equipped with a special hardware such as wake-up radio. Also, using a wake-up radio to activate transceiver forces all nodes within one-hop range turn on their transceiver regardless of what the radius of the Frisbee is set. Moreover, having always-on nodes are not feasible in energy-constrained wireless networks and, therefore, not applicable to wireless sensor networks. Having only two types of modes (i.e., active or not) in terms of the level of activeness also limits the adaptability of the network to dynamically changing environments.

Another approach described in [26] adopts a modified Frisbee model for a surveillance application in a random sensor network. The surveillance system is structured by a static cluster-based tracking architecture consisting of two types of sensors: Simple sensors that have only sensing capability are pre-determined as cluster members while complex ones that have computation capability as well serves as cluster heads. A target from a sensor is first localized using Maximum Likelihood estimation in cluster members and then combined using Kalman filter in a cluster head. Once the location of the target is estimated, then an asleep-awake mechanism is employed for energy-efficient operations, which selects a set of nodes that needs to be activated. The mechanism defines a model with two active zones and lets the nodes within the zones fully active. The first active zone is defined as the circular region centered at the estimated current location of the target being tracked, which is exactly same as the Frisbee model. To better help the target tracking in case of large errors, the second active zone is defined, which is centered at the estimated target track in the

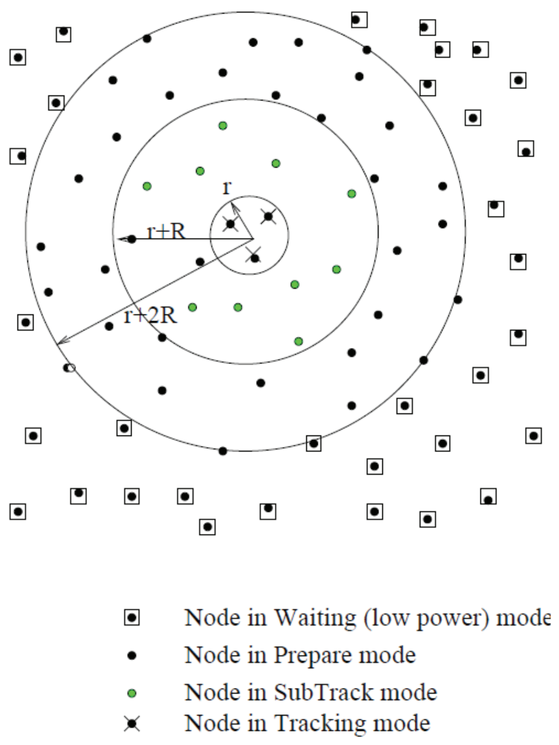


Figure 3.11.: Layered onion-like node state distribution around the target in PECAS (reproduced from [43])

previous recursive step. Since the active zones are maintained as a new zone with a single lag, this model is called “Frisbee model with memory”.

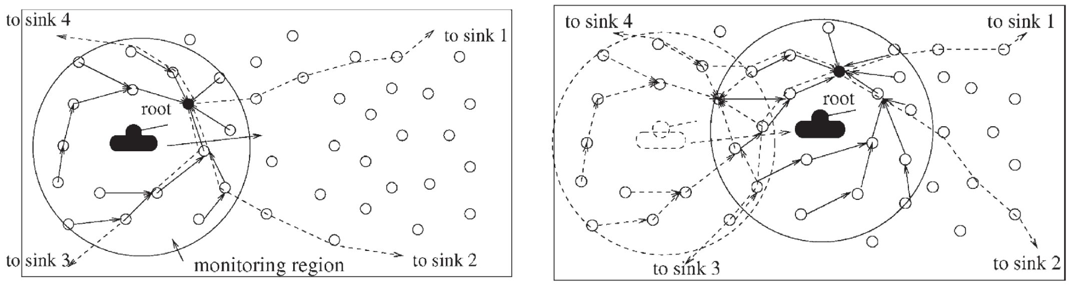
Although this approach improves the Frisbee model in terms of tracking, it still inherits the shortcomings of the original version except for the always-on node case. In addition, the aforementioned approaches seem to somewhat take into consideration the future state of the target of interest, yet based heavily on a heuristic reasoning.

Probing Environment and Collaborating Adaptive Sleeping (PECAS) described in [43] is based on a similar rationale in a sense that only a subset of the network around an event of interest needs to be active for energy efficiency. Whereas the nodes in the aforementioned approaches are awakened by other nodes using a external wake-up signal, however, the nodes within the active region in this approach broadcast a wake-up *packet* so as to proactively wake-up other nodes. There are four states defined as illustrated in Figure 3.11: the nodes in Tracking mode are the nodes that actually detecting an object and participating a collaborative tracking, which are in the region of event occurring. The nodes in SubTrack mode are the nodes that can *overhear* the packets from the nodes in Tracking mode, which are within one-hop communication range. Each node in SubTrack mode broadcasts a Prepare message so as to wake-up another set of nodes, which are in the region of two-hop communication range. Thus, the nodes in Tracking, SubTrack, and Prepare modes within two-hop range are in fully active state. Other nodes will be in a low-power mode.

This type of proactive wake-up mechanism makes the network adaptive and prepared for upcoming possible events; however, a large amount of wake-up packets generated in a local area may cause a huge communication overhead and enhance congestion around an object, resulting in increased packet collisions and hence may degrade tracking performance. More energy consumption is also obviously entailed.

While the approach in [44] operates nodes in a similar way, it has a different method to determine the range of adjusting the duty cycle around the event of interest. The method dynamically changes its duty cycle based on the number of hops from the event-detecting node. Also, its duty cycle adjustment is performed *additively* so that the sleep interval changes in, for example, $1T$, $2T$, $3T$, and $4T$, whereas the DSMAC and AMAC approaches take exponential duty cycle adjustment, for example, $1T$, $2T$, $4T$, and $8T$. However, the downside of an additive adaptation of duty cycling is already pointed out in [7, 8] that it is hard for a node to synchronize with its neighbors since if the neighbors have different schedules, then there may not be a single overlapping SYNC slot for all the neighbors, resulting in multiple transmissions of a SYNC packet by the node.

Instead of waking up all the nodes within a fixed range, the approach described in [45] *predicts* a smaller set of nodes that an object may appear in the next time step, and then wake them up only. To make prediction, it first stores the object movement log over time in a type of database and then discover the movement pattens of objects by mining the database, generating object movement rules. Then a region that an object is likely to appear in the future is predicted, and the nodes within the region



(a) (b)

Figure 3.12.: Using convoy tree to track the target. (a) Data collection (b) Tree reconfiguration (reproduced from [46])

will be activated. Since the network is hierarchically structured with multiple-level clusters, if an object is lost by the failure of the prediction, the nodes within a larger region defined in a higher level will be activated by the cluster head until the missing object is found.

Since the entire past history of objects that appeared is stored and mined, the prediction would work well if the type of objects and the movement patterns are relatively static rather than random. Therefore, although this approach indeed predicts the next nodes that are likely to detect an object, its usage is limited and so not suitable for dynamic networks in terms of the target's mobility.

There are also other node activation schemes in a structured network. Convoy-tree approach proposed in [46] is based on a tree-based reconfigurable structure and adaptively activate/deactivate nodes using Geographical Adaptive Fidelity (GAF) [47], as illustrated in Figure 3.12. Collaborative tracking operation in this approach entails tree expansion and pruning with regard to the target's speed.

Above all, it is not clear what the aforementioned approaches mean by activating nodes. Even when they define activation as turning on radio or increasing duty cycle, they fail to address how to successfully perform the activation/deactivation through communications among the nodes that may have heterogeneous schedules. If they do so in a naive way that a node retransmits a packet if unsuccessful, it may incur multiple retransmissions, resulting in energy waste and possibly poor throughput/latency

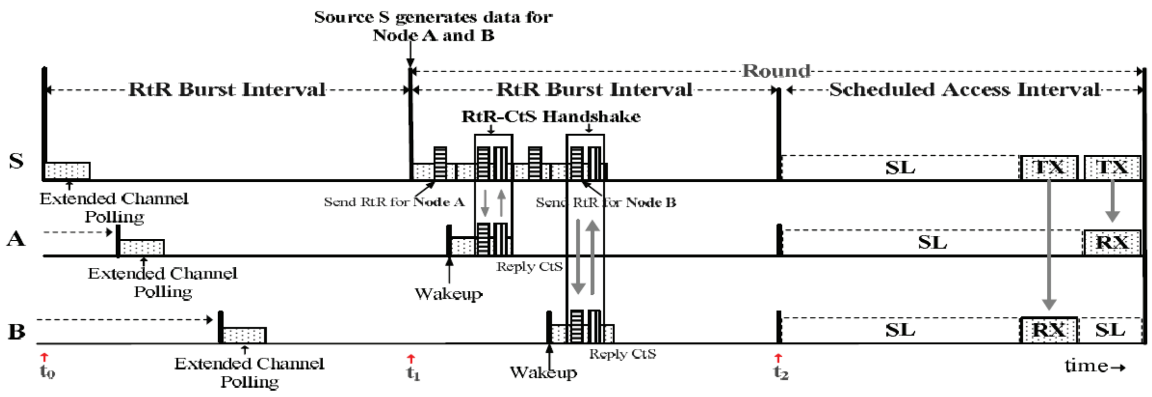


Figure 3.13.: A timeline of a sender (Node S) and two receivers (Node A and B) in EC-MAC (reproduced from [46])

due to enhanced contention. Without stating how to actually schedule communications in different modes, therefore, it is difficult to imagine how to get performance gain.

EC-MAC (Event-Centric MAC) [42] points out that all the existing works inspired by the Frisbee model handle the issues on selective activation of a subset of nodes related to an event in only either application or network layer. Instead, EC-MAC provides a hybrid MAC scheme of asynchronous random access and TDMA that can facilitate efficient and reliable communication in a cluster-based collaborative data processing and forwarding. In an idle state, all nodes operate with asynchronous duty cycle schedules. When an event of interest occurs, a burst of RtR (Request to Receive) packets is initiated by a node and wakes up all the nodes within an active region and set up/reserve a TX-RX exchange schedule with intended receivers, as shown in Figure 3.13. Corresponding receiver nodes reply back with a CtS (Clear to Send) packet which is similar to the early ACK in X-MAC. Using this scheme of medium access control, a tree-based cluster is constructed among the nodes that are currently observing the event. The root node (i.e., cluster head) then assigns time slots to its one-hop neighbors. These neighbors then propagates the schedule to the leaf nodes that may be in multi-hop away from the root node.

While EC-MAC provides an event-driven local data collection mechanism among asynchronous schedules, its objective is set to reliably and efficiently collect data within an active region. Therefore, its design does not take into account the delay-sensitivity of generated traffic. In addition, for the methods inspired by the Frisbee model, it is crucial how an active region is defined and what should be the radius of it, because it affects the level of preparation of the nodes for the upcoming events. However, EC-MAC defines the active region as only the set of nodes that are *already* detecting an event, which contradicts to the motivation of the Frisbee model. Moreover, it assumes to use an existing tree construction algorithm which may greatly affects on the performance of the data collection in terms of latency.

Note that all adaptive approaches in network protocols and all predictive approaches for network adaptation are based on the proximity of the actual nodes to the event. This implies that all of them are designed under an assumption that their target wireless sensor networks employ non-directional sensors, the center of the sensing field of which is the location of the sensor node. Since camera is a highly directional sensor (it may have very narrow field of view, causing strong directionality), their assumption does not hold and thus they are not suitable for wireless camera networks. Unless all cameras are deployed in a way that they are viewing strictly downward, which is obviously impractical assumption, the predictive approaches for network adaptation in wireless camera networks must be based on the distance metric between the event and the sensing region of the nodes.

(Paul: TODO: Include [27])

3.4 Tracking Approaches for MAC Layer

The aforementioned predictive approaches for network adaptation are mostly based on a heuristic reasoning. Making a statistically optimal prediction is obviously preferred for network adaptation, and it must be entailed by employing an optimal estimator with given measurements such as Kalman filters and Particle filters. Suppose we implement a tracker in the MAC layer to get optimal prediction. Then we should not assume that accurate measurements are available. Oftentimes what a node can infer in the MAC layer by overhearing on-going packets is whether an object of interest is present or not, which is embedded in the packet header as a binary information. Thus, in the viewpoint of MAC layer, we can think of other nodes as binary sensors regardless of the resolution of their actual sensors. Even if a couple of bits are used to represent the state about an object, it must be in an extremely low resolution. Therefore, it is important to investigate how a tracking task is performed with binary sensors in a sensor network context.

Tracking objects using binary sensors in a large-scale real sensor network deployment is successfully demonstrated in [48]. The tracking methods or trajectory estimation strategies among literature for binary sensor networks roughly bifurcated as piecewise linear path approximation-based [49–51] and particle filter-based approaches [50, 52, 53].

While the former is simple and computationally inexpensive, we are more interested in the latter approaches which are based on Particle filter [54–56] since they provide us optimal estimation, and recent advances in embedded systems such as Imote2 [57] allow us to employ more sophisticated algorithms with less cost and energy.

The authors in [50] propose two minimal trajectory estimations and representations with the ideal and non-ideal binary sensing models, respectively. For the ideal sensing model where the sensing range of a binary sensor is modeled as a circular region with a fixed radius, a minimal set of connected line segments is identified that pass through all arcs of sensing boundaries belonging to a localization patch as shown in Figure 3.14. A localization patch is defined as the intersection of the sensing range of the nodes that are simultaneously detecting an object of interest at a given time instant, subtracted by the union of the sensing range of the nodes that do not detect the object at the same time instant. In 2D case, for example, it is analogous to a tessellation. For the non-ideal sensing model where a uncertainty region lies

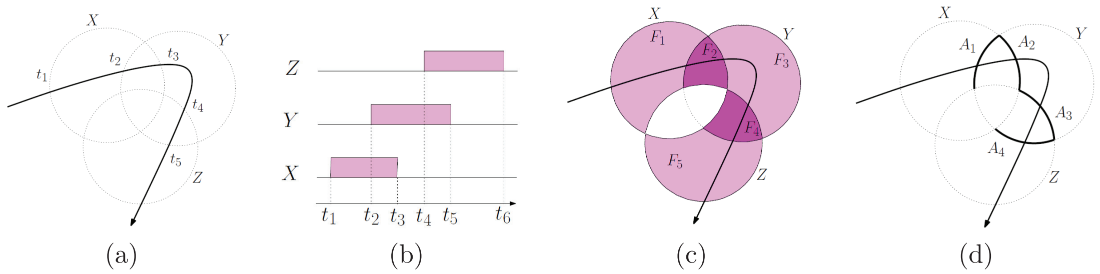


Figure 3.14.: The minimal trajectory estimations and representations with the ideal binary sensing model: (a) Shows a target moving through a field of three binary proximity sensors, X, Y, and Z (b) Shows sensor output as a function of time (c) Shows the localization patches to which the target is localized (d) Shows the arcs marking boundaries between patches (reproduced from [46])

outside the ideal sensing range, the piecewise linear approximation yields a poor performance in terms of trajectory estimation. Therefore, a particle filter is employed to accommodate this non-ideality in tracking. After estimating the trajectory using the particle filter, a geometric piecewise linear approximation algorithm is applied to get a minimal trajectory representation as a post-processing.

The particle filtering for non-ideal sensing model in [50] is summarized as follows: At an initial state, it is assumed that we have K particles associated with the current location of a target at a time instant n: $x_k[n]$. At the next time instant $n+1$, assuming that an object is detected at the localization patch F, draw m candidate particles randomly for each particle $x_k[n]$ uniformly over the patch F, resulting in total mK candidates. Then, we choose a new set of K particles among mK candidates that minimize the cost function defined as the sum of the total acceleration between time instances over the history of the target. This type of cost function yields the lowest-pass trajectory evolution since the high frequency part that corresponds to high acceleration is minimized.

This approach, however, requires a number of strong assumptions in terms of communication. The basic assumptions include the perfect time synchronization among nodes and the relatively accurate localization of nodes. Moreover, a tracker node—

which corresponds to the cluster head in a cluster-based tracking system—is assumed to be able to collect all sensor observations without any communication issues. In addition, the identification of the localization patches is a critical step in tracking objects. Yet it must be predicated by the fact that a tracker node has sensor readings from its neighbors with the exactly same time stamp. In practice, however, it is usually very hard to make these assumptions. Furthermore, if communication latency is not negligible, which is actually true in most event-driven collaborative wireless sensor networks, these assumptions will no longer hold. Nonetheless, the investigation on the fundamental limit of spatial sensing resolution in tracking using binary proximity sensors of this work provides us a useful theoretical analysis to better understand tracking performance in a binary sensor network, albeit in a high level view with an idealized communication setting.

The work described in [53] improves the above approach in a sense that multiple target tracking in a binary sensor network is enabled with theoretical analysis on the target countability. To track multiple targets, the same particle filter technique described above is used yet with clustering of particles. This particle cluster-based target identification is essentially based on the intuition that if there are multiple targets, then particles will form multiple clusters around the center of the targets. Therefore, the number of clusters in particles will correspond to the number of targets within a particular localization patch at a given time. This approach is an improved version of the previous approach yet inherits all the shortcomings, too.

Instead of identifying localization patches, the particle filter employed in [52] for tracking with binary sensors draws a set of random samples from the entire sensing region of a binary proximity sensor detecting an object. The goal of this approach is to estimate the direction of the movement of the object of interest as well as the current location. For object direction estimation, it requires a type of binary sensors that output whether an object is approaching to a node or moving away from it. For object localization, on the other hand, it requires another set of binary proximity sensors, resulting in multi-modality in sensor network deployments. Without the proximity information, they show that there exists indistinguishable pairs of trajectory for any binary sensors. Since this approach requires special types of binary sensors for target localization, it is not generally applicable especially to tracking task in MAC layer.

In addition to the aforementioned centralized particle filters with binary sensors, there are distributed versions of particle filters used in general sensor networks, which can be found in [58, 59].

4. OBJECT TRACKING FROM THE BOTTOM OF NETWORK

In this section, we present a novel framework for object tracking from the bottom of 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 4.1. The communication range and the field-of-view of each camera node are shown, respectively, in Figures 4.1(a) and (b). The goal of this camera network is to track an object of interest depicted as the star in Figure 4.1(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 4.1(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 6.1.4. 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 6.1.5.

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

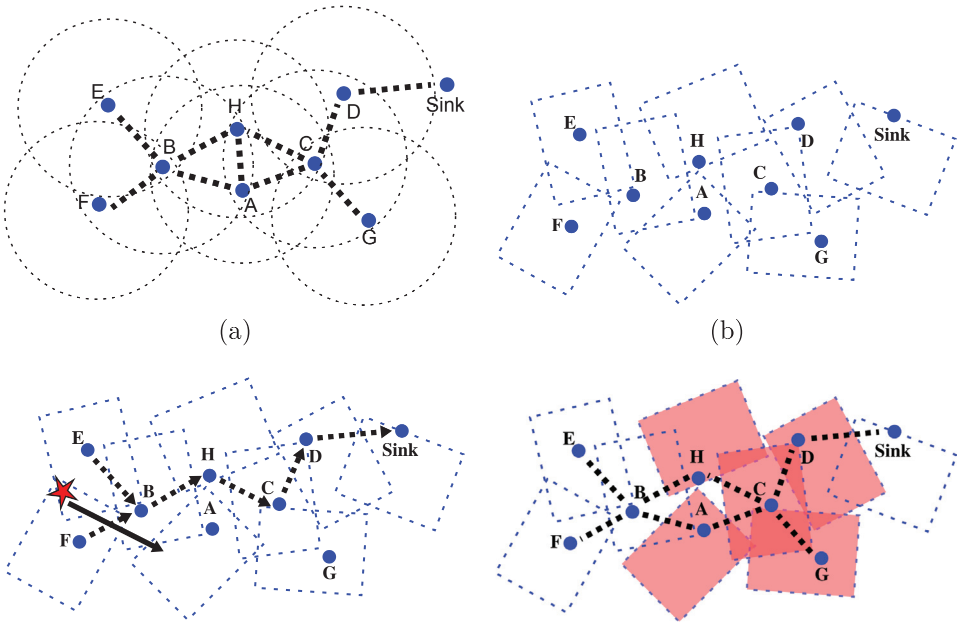


Figure 4.1.: A depiction of a WCN engaged in tracking a moving object. The dotted circles and lines in (a) represent the communication range and connectivity among nodes, respectively, and the dotted rectangles in (b)-(c) represent the sensing field of nodes. The red star in (c) indicates the object 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.

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 4.1(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

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

network have been localized and calibrated and that the field-of-view of each camera is available in the form of a 3-tuple $(i, \mathbf{z}, R)$ where i identifies the node (i_{self} is the local node address), $\mathbf{z}$ 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 PNAT 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. 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 [54,58], but we chose to use Kalman filter [60] 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 PNAT 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

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T .$$

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

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

and $\mathbf{w}_k = [\ a_x \quad a_y\ ]^T$ 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

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

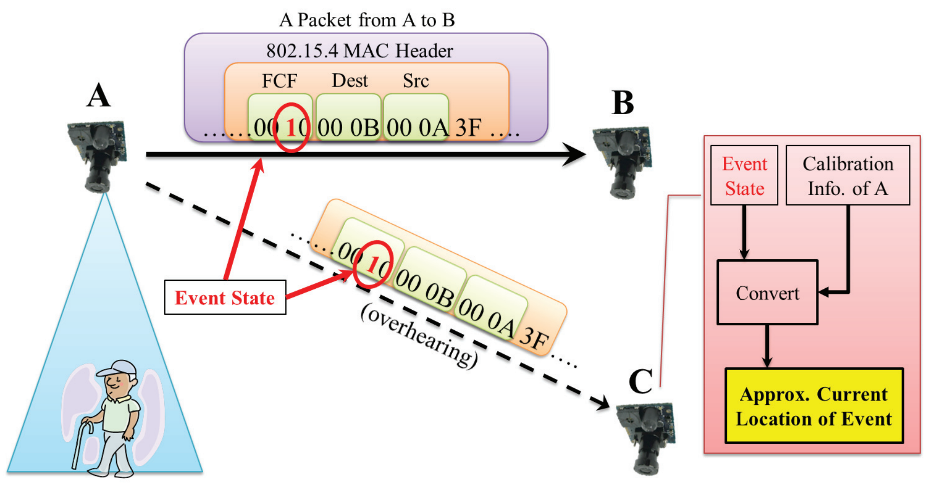


Figure 4.2.: An illustration of how the event packet detection is used not only to get advance notice of an approaching target but also to decide whether the target is close enough for any changes to the local communication parameters. Node A is currently detecting an object and wants to collaborate 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.

where $H_{k+1} = \left[\begin{array}{cc} I_{(2\times2)} & 0_{(2\times2)} \end{array}\right]$ 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

$$\hat{\mathbf{x}}_{k+1|k} = F_k \hat{\mathbf{x}}_{k|k} \tag{4.1}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + W_k Q_k W_k^T . \tag{4.2}$$

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

Algorithm 4.1 Object tracking via indirect sensing using KF in the tracker module: Cross-layer interaction from MAC and application layer

< Application layer >

```
 1: $\mathbf{z}_{direct} \triangleq$ a measurement from camera in a form of coordinates
 2: $R_{direct} \triangleq$ the covariance matrix associated with the measurement $\mathbf{z}_{direct}$
 3: loop                                          ▷ Infinite loop while a node is running
 4:     if a new sensing result is available then
 5:         if an object is detected within the sensing field at time $t_k$ then
 6:             Provide tracker module the new measurement $(\mathbf{z}_{direct}, R_{direct})$
 7:         else                                                ▷ No object is detected
 8:             Provide tracker module a $NULL$ measurement
 9:         end if
10:     end if
11: end loop
```

< MAC layer >

```
 1: $packet(i) \triangleq$ a packet transmitted by node $i$
 2: $NodeId[Size] \triangleq$ an array to store the list of node IDs
 3: $isDetecting \triangleq$ a boolean indicating whether the node is currently detecting an
    object or not
 4: $T_{timeout} \triangleq$ the periodic time interval to send $NodeIDs$ to tracker module
 5: A timer is set to expire at every $T_{timeout}$
 6: loop                                          ▷ Infinite loop while a node is running
 7:     if a new $packet(i)$ is available with EEN bit set then      ▷ Indirect sensing
 8:         Insert $i$ into storage $NodeId$
 9:     end if
10:     if timer is expired || $NodeId$ is full then
11:         Send storage $NodeId$ to tracker module and Reset timer and storage
    $NodeId$
12:     end if
13:     if a packet is ready to be sent && channel is clear then
14:         EEN bit of the packet $\leftarrow isDetecting$
15:     end if
16: end loop
```

Algorithm 4.2 Object tracking via indirect sensing using KF in the tracker module: Cross-layer interaction at the tracker module

< Tracker Module >

```
1: $\mathbf{z}_{indirect}(i) \triangleq$ the center of the sensing field of node $i$
2: $R_{indirect}(i) \triangleq$ the ellipsoidal approximation of the sensing field of Node $i$ represented as a covariance matrix
3: loop                                              ▷ Infinite loop while a node is running
4:     if an array $NodeId[N]$ is received from MAC layer then ▷ Indirect sensing
5:         for $n := 0$ to $N - 1$ step 1 do
6:             $Kalman(\mathbf{z}_{indirect}(NodeId[n]), R_{indirect}(NodeId[n]))$    ▷ calls Function
7:         end for
8:     else if a new $(\mathbf{z}_{direct}, R_{direct})$ is received from application layer then      ▷ Direct sensing
9:         if $\mathbf{z}_{direct} \neq NULL$ then
10:            Set $isDetecting \leftarrow True$ in the MAC layer
11:            $Kalman(\mathbf{z}_{direct}, R_{direct})$                       ▷ calls Function
12:        else Set $isDetecting \leftarrow False$ in the MAC layer
13:        end if
14:    end if
15: end loop
```

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 4.2 illustrates this process.

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 [61]. 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 4.1 and 4.2 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

$$S^j_{k+1|k}(\mathbf{u}) = Pr(\mathbf{p}^j_{k+1|k} = \mathbf{u}). \tag{4.3}$$

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. (4.3).

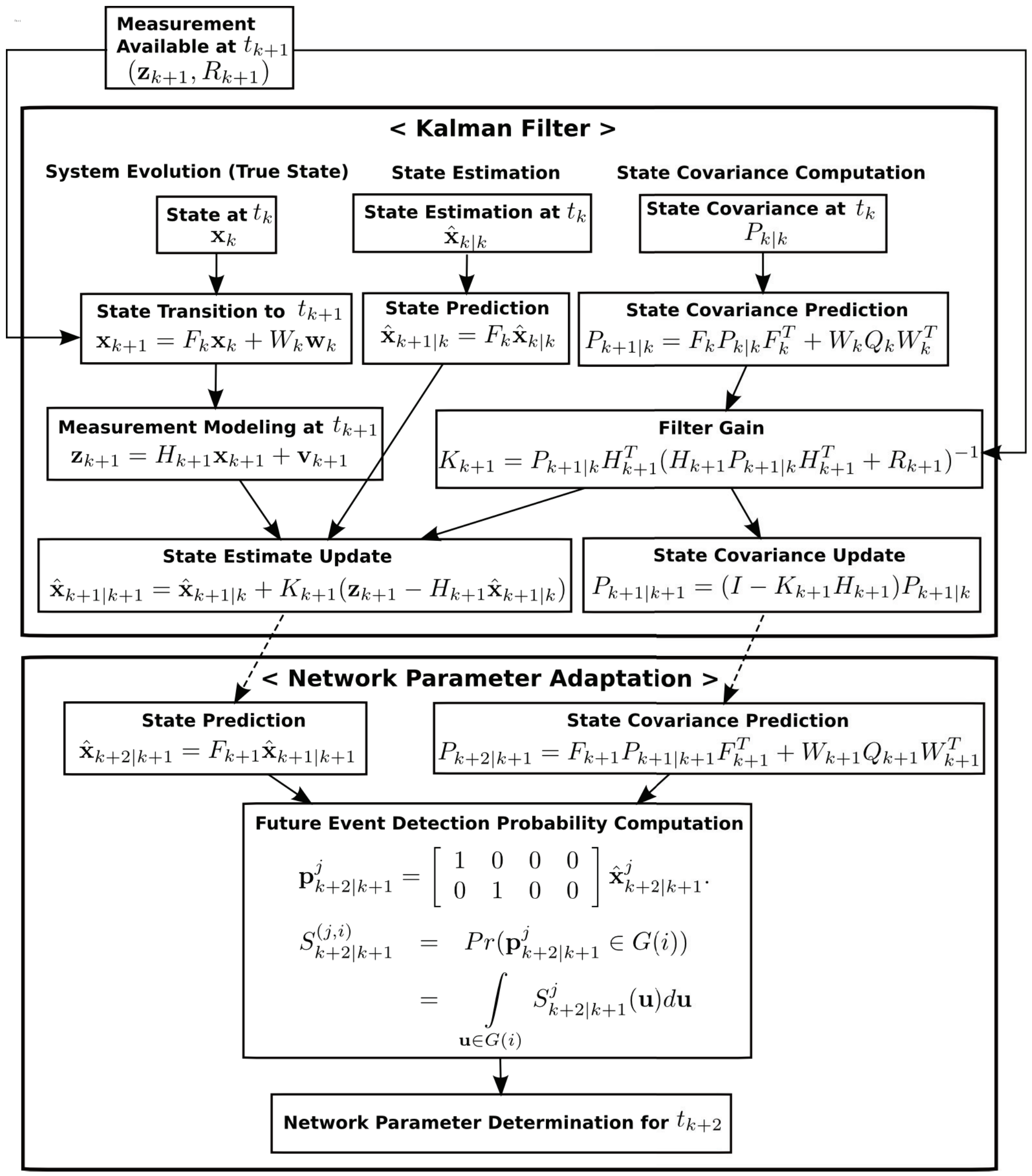

Measurement Available at t_{k+1} $(\mathbf{z}_{k+1}, R_{k+1})$
< Kalman Filter >
System Evolution (True State)
State Estimation
State Covariance Computation
State at t_k $\mathbf{x}_k$
State Estimation at t_k $\hat{\mathbf{x}}_{k|k}$
State Covariance at t_k $P_{k|k}$
State Transition to t_{k+1} $\mathbf{x}_{k+1} = F_k\mathbf{x}_k + W_k\mathbf{w}_k$
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}

Figure 4.3.: The state transition diagram that describes the steps to determine a new network parameter value based on FEDP.

5. PREDICTIVE NETWORK ADAPTATION BY TRACKING

While the network protocol stack keeps track of a target object by the tracker module utilizing the event information from the bottom of the network, we would want the adaptation of the network parameters to take place simultaneously in the individual protocol layers connected to the tracker module. For a network layer to be able to adapt its parameters, it is necessary to translate the estimated state of an object of interest into a quantity based upon which the adaptation is carried out. In this chapter, we first describe the overall system architecture and then introduce two metrics that are used for converting the currently estimated state of the object into a proper level of parameter that would need to be adopted in the future.

5.1 Overall System Architecture

The proposed framework, which we will refer to as *Predictive Network Adaptation by Tracking* (PNAT), can be applied to any network systems with synchronous MAC protocols where a sender and its potential receivers share the same active periods. Figure 5.1 shows how the framework 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 PNAT 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 PNAT-enabled network

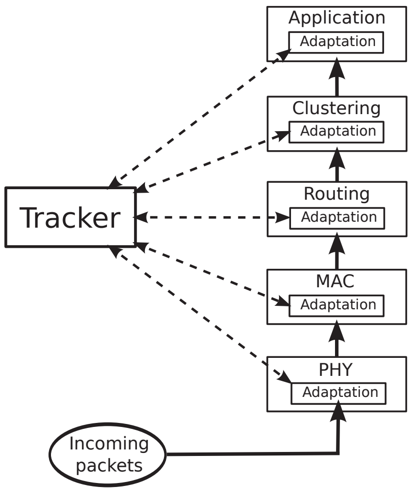
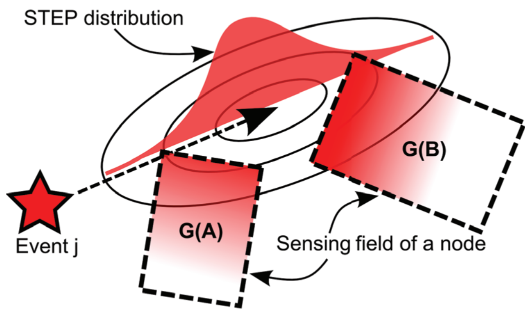


Figure 5.1.: Overall system architecture of PNAT: The tracker module is implemented separately, not belonging to any layer within the protocol stack. The adaptation modules reside in individual layers since each of them is responsible for adapting different network parameters in different layers. The 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.

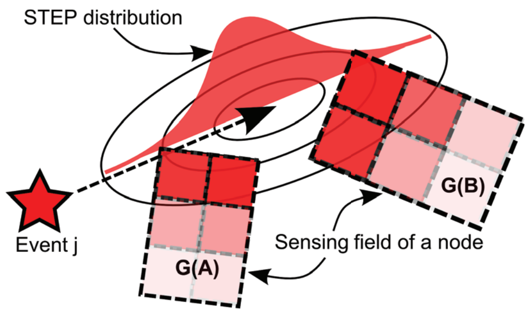
can achieve high energy efficiency and tracking performance. Such synergy is created when each node in the camera network performs both object tracking and network parameter adaptation simultaneously.

5.2 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).



(a)



(b)

Figure 5.2.: STEP distribution for an event j and FEDP of Node A and B: **(a)** The FEDP value of Node A and B, $S_{k+1|k}^{(j,A)}$ and $S_{k+1|k}^{(j,B)}$, are the integration of STEP, $S_{k+1|k}^{j}(\mathbf{u})$, 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.

5.2.1 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:

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

where $G(i)$ denotes the sensing field of the node i as illustrated in Figure 5.2(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 4.3 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 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.

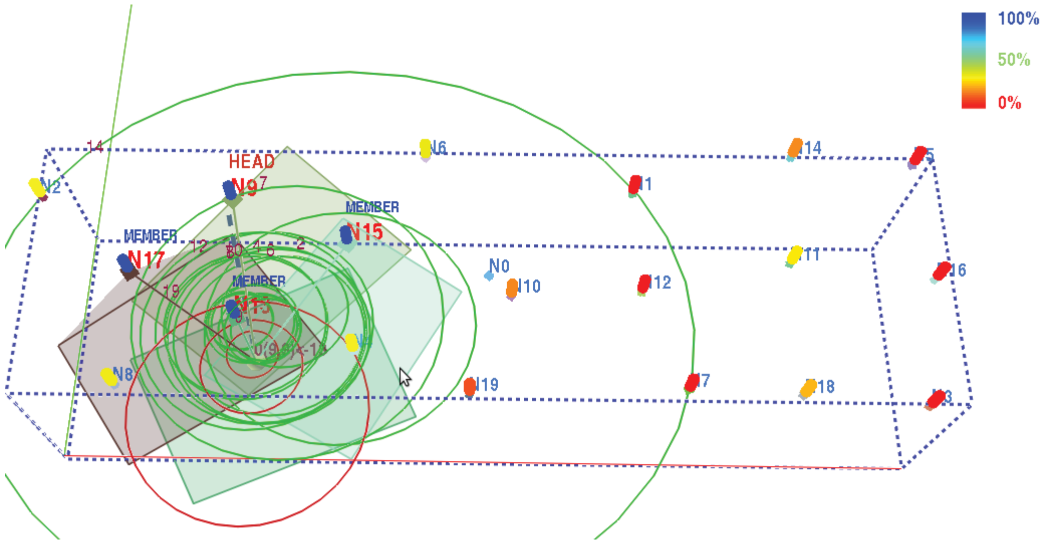


Figure 5.3.: Tracking by Kalman filter at each node. Small and large red circles indicate the current and the predicted positions of an object as estimated by the Kalman filter at the current cluster head. Small and large green circles are those estimated by the Kalman filter at each of non-cluster nodes. The color of the node indicates the probability that the object will enter its field-of-view in the near future (4 seconds).

5.2.2 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.

Figure 5.3 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 5.3, 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 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.

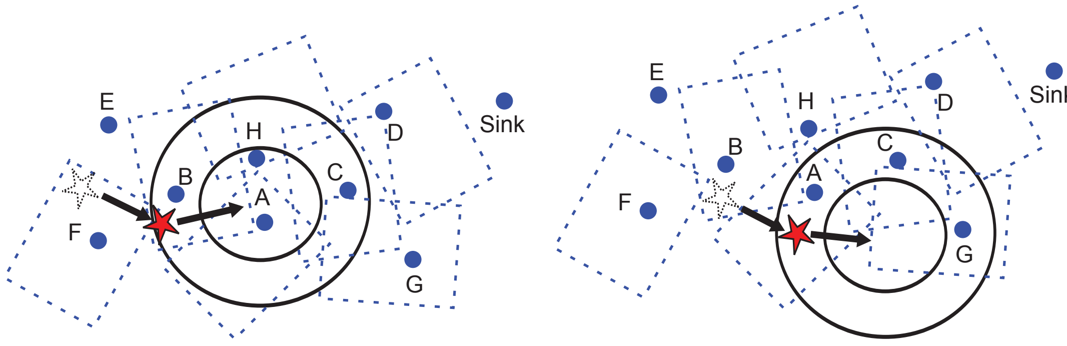


Figure 5.4.: The continued depiction of a WCN engaged in tracking a moving object at two subsequent time instants. The red star indicates an event and the black solid arrows its moving direction. The regions divided by black solid circles indicate examples of a contour map of the STEP of the event predicted by a node.

6. SIMULTANEOUS MULTI-LAYER NETWORK ADAPTATION

Radio is obviously one of the biggest energy consumer in wireless sensors including wireless cameras, and the control of radio operation and the scheduling of packet transmission is governed by the MAC layer protocol employed. Adding the PNAT to the MAC layer, it is made possible for a node to adapt its radio duty cycle even before an object of interest appears in its field-of-view — which will be referred to as the *Predictive Duty Cycle Adaptation* (PDCA) — while the node is tracking the object beyond its own field-of-view, and thus a significant energy saving can be made in radio operation while supporting the QoS requirements to the application.

Yet as we described earlier, collaborative visual processing entails extremely high energy expenditure, necessitating tackling energy efficiency in multiple layers and multiple components simultaneously, especially for those components who consumes energy most. In addition to minimizing the energy consumption at radio, therefore, it is critical to also minimize the energy consumption at the camera and in all the image processing modules from image acquisition to high-level vision processing since it requires intensive computation. Table 6.2 summarizes the energy expenditures of different hardware components of Imote2 mote [57] in different operation modes. Adding the PNAT to the application layer, therefore, the camera sensing rate in the application layer can be adapted in advance before an object of interest appears in its field-of-view — which we will refer to as the *Predictive Sensing Rate Adaptation* (PSRA).

In this chapter, we first show two examples of how an individual layer adapts its parameters by consulting the event tracker on how likely an object of interest would appear in the field-of-view in the near future: one in the MAC layer (i.e., PDCA) and the other in the application layer (i.e., PSRA). Note that multiple layers can adapt their parameters simultaneously, and in the next chapter we will show the energy saving in two cases when PNAT is applied to a single layer (i.e., PDCA only) and multiple layers simultaneously (i.e., PDCA + PSRA), respectively.

6.1 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.

6.1.1 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.

Consider an example illustrated in Figure 5.4 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 5.4(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 5.4(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 5.4(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 5.4 (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

Table 6.1: Parameters and symbols used

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

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 6.1. The definitions of parameters used are summarized in Table 6.1.

Algorithm 6.1 Duty cycle adaptation

```
1:  loop                                              ▷ Infinite loop while a node is running
2:      if an object exists within the sensing field then              ▷ Direct sensing
3:          Schedule to adopt the highest duty cycle level $d_N$ at the earliest time slot
4:      else if current time is almost scheduled time then
5:          Compute STEP $S^{j}_{k+1|k}(\mathbf{u})$ for event $j$ and FEDP $S^{(j,i)}_{k+1|k}$ for a node $i$
6:          Determine its proper duty cycle level $d_m$ based on FEDP
7:          Schedule/reschedule to adopt the new duty cycle level $d_m$ at $t_{k+1}$
8:      end if
9:      if $d_c \neq d_m$ then
10:         Set $d_m$ to be the current duty cycle level with a timeout
11:     else                                                            ▷ $d_c == d_m$
12:         Refresh timeout of $d_c$
13:     end if
14:     if a new schedule is adopted then
15:         Broadcast the new schedule to neighboring nodes
16:     end if
17: end loop
```

6.1.2 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 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.

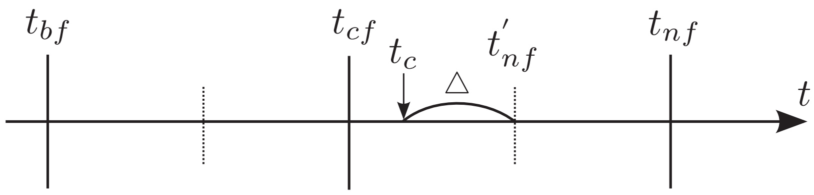


Figure 6.1.: Timeline for adopting a schedule with a higher duty cycle.

6.1.3 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

$$\Delta = min\left[U\left(\frac{T_0}{M^{d_m}}\right) - (t_c - t_{bf})\right], \tag{6.1}$$

subject to $\Delta > 0$, where $U \in \{1, \ldots, M^{d_m}\}$. Since all the parameters in Eq. (6.1) 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 6.1. 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. (6.1), and the beginning time of the next frame $t^{'}_{nf}$ is rescheduled accordingly.

6.1.4 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 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 6.2(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:

$$f(i+1) = (f(i) + M^{(d_{N-1}-d_c)})\, mod\, M^{d_{N-1}} \tag{6.2}$$

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 6.2(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:

$$f(i)\, mod\, (M^{d_{N-1}-d_c^{peer}}) == 0. \tag{6.3}$$

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 (6.3) 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 (6.3) 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.

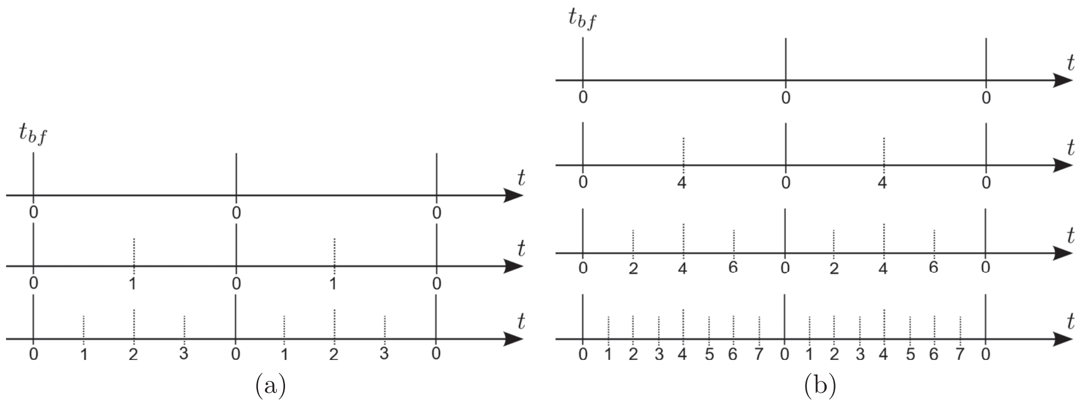


Figure 6.2.: (a) An example of inconsistent frame numbering, when $M = 2$ and $N = 3$. (b) An example of consistent frame numbering, when $M = 2$ and $N = 4$. From the top timeline, d_c corresponds to 0, 1, 2, and 3, respectively.

6.1.4.1 Neighbor Synchronization

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

Algorithm 6.2 Support for fast delivery of routing packets related to an object of interest.

```
 1: $p_{in}(i) \triangleq$ a packet received from node $i$ via unicast        ▷ Intended recipient
 2: $p_{out}(i) \triangleq$ a packet to be transmitted to node $i$
 3: $isEventRouting \triangleq$ a flag indicating whether a node is part of a routing path of
    event-related packets
 4: loop                                        ▷ Infinite loop while a node is running
 5:     if a new $p_{in}(i)$ is detected with EEN or EERN bit set then
 6:         Set $isEventRouting$ with a timeout
 7:         $d_m \leftarrow d_{routing}$
 8:         Schedule/reschedule to adopt the new duty cycle level $d_m$
 9:     end if
10:     if a packet $p_{out}(i)$ is ready then
11:         $p_{out}(i).EERN \leftarrow isEventRouting$
12:     end if
13:     if timeout expires then
14:         Unset $isEventRouting$
15:         $d_m \leftarrow d_0$
16:         Schedule/reschedule to adopt the new duty cycle level $d_m$
17:         Broadcast the new schedule to neighboring nodes
18:     end if
19: end loop
```

Table 6.2: Energy expenditures at different hardware components of Imote2 [62, 63]

Components	**Mode**	**Power (mW)**
Imote2 w/o radio	Active	192.4
Imote2 w/o radio	Idle	156.4
Camera	Active	72.0
Camera	Idle	26.4
Radio	Active	82.4

6.1.5 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 4.1(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 6.2. In the algorithm, the variable *isEventRouting* indicates whether a node is currently a part of a routing path.

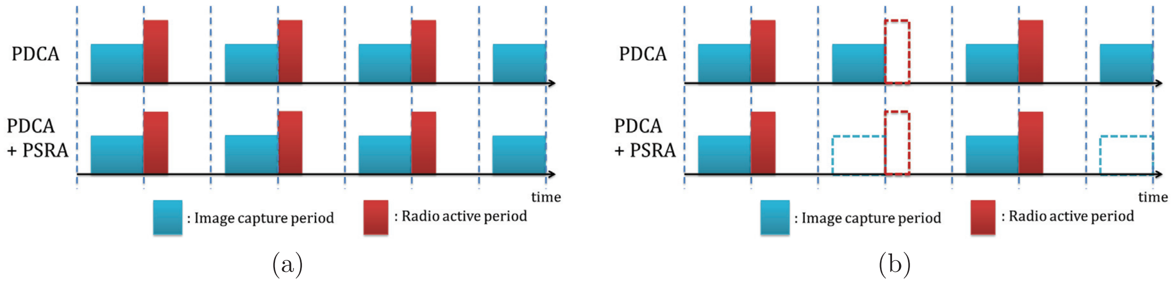


Figure 6.3.: Adaptive scheduling of radio and sensing: The time periods colored as red and blue indicate the active periods of radio and the period of the image processing, respectively. The schedule of sensing and radio activation is illustrated when the FEDP is high and low in (a) and (b), respectively.

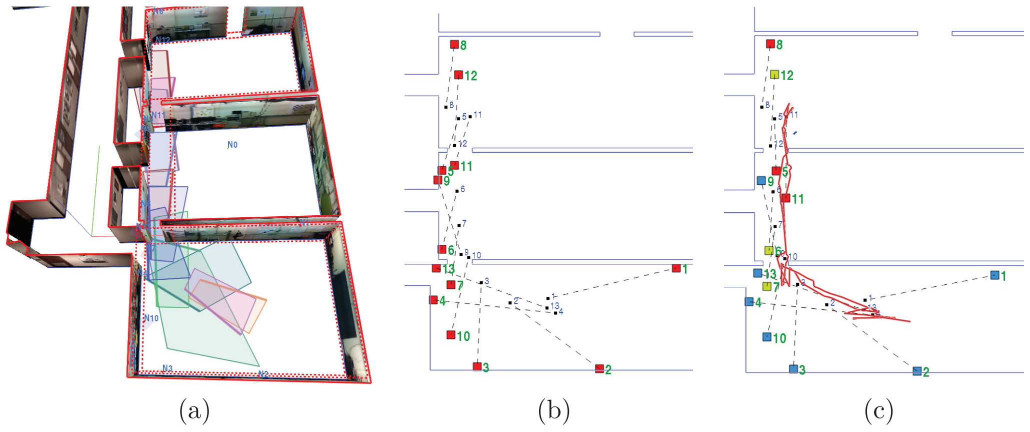


Figure 6.4.: The Purdue RVL Wireless Camera Network Testbed used for the performance evaluation consists of 13 Imote2 motes with cameras deployed along the doorway across three rooms: (a) A 3D model of the testbed with the field-of-view of each camera depicted as a colored polygon on the floor; (b) A plan view of the testbed with the physical location of each camera drawn as a small red box and the center of its field-of-view drawn as a small black box connected to the red box with a dotted line; and (c) An example of the trajectory of a mobile object estimated by the testbed.

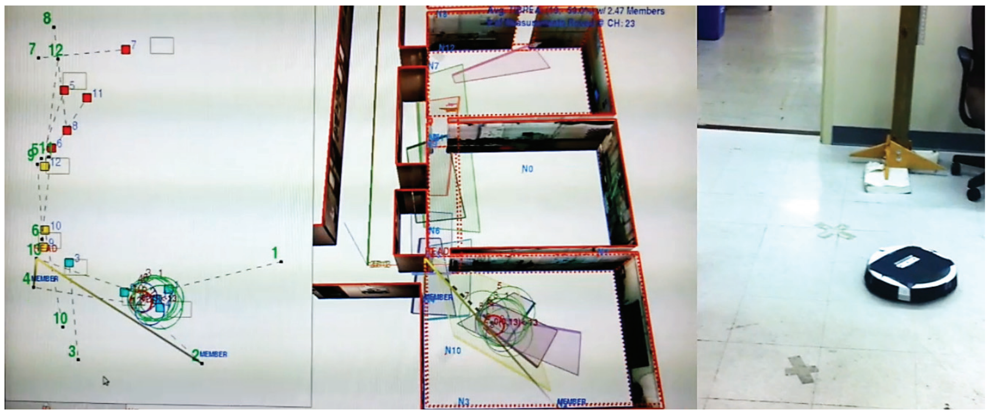


Figure 6.5.: A snapshot of a real object and the GUI-based visualization of the object tracking by the Purdue RVL WCN testbed: The plan view and the 3D model of the testbed are in the left and in the middle, respectively, while tracking a real mobile object on the floor as shown in the right.

6.2 Predictive Sensing Rate Adaptation

The event tracker implemented as a separate module along with the protocol stack is designed to be connected by each individual layer including the application layer. The camera management system in the application layer can consult with the event tracker and thus control the camera sensing rate efficiently in a predictive manner while providing application-level QoS. We refer to this as *Predictive Sensing Rate Adaptation* (PSRA) and present in this section how the PSRA is carried out by taking advantage of the event tracker.

Provided the FEDP or EToE estimated in the tracker module at a node, each node can choose a proper value for its sensing rate in the application layer. A higher FEDP or a lower EToE would cause a node to choose a larger value for its sensing rate since the camera needs to watch the environment with more caution. In a cluster-based object tracking application, the maximum sensing rate at a node would be determined by taking into account both the data aggregation rate at the cluster head and the maximum sensing rate that the hardware supports. If the data aggregation at the cluster head takes place at every second, for example, the cluster member does not need to capture images faster than this although the hardware allows capturing images at a higher rate.

The minimum sensing rate, on the other hand, would be determined based on what the application allows as to how much it would be tolerable in terms of the network-level and node-level detection delay. Note that the network-level and the node-levl detection delays are different in that the network-level detection delay counts how far an intruder, for example, can reach inside the sensing coverage of the entire network while the node-level delay does so for the sensing coverage of an individual node. The network-level detection delay is usually more tolerable than the node-level one, since the network-level detection delay is just an one-time delay for an intruder or a new object even for tracking applications while the node-level delay keeps contributing to the overall performance of the cluster-based tracking application as a new node detects the object and joins to the existing cluster.

Note that using the proposed PSRA approach, each node is allowed to set its minimum sensing rate corresponding to application-specific requirement in terms of the network-level detection delay, since once an object of interest appears within the sensing coverage of the network, the PSRA allows each node to be alerted in advance and sets its sensing rate appropriately if the object is likely to appear to the node in the near future. If the sensing rate adaptation is carried out in a reactive manner, on the other hand, the minimum sensing rate must be bounded by more stringent node-level detection delay than the network-level detection delay.

As long as there is a gap between the lower and the upper bounds of the sensing rate, it should be possible that the energy consumption can be further minimized by adapting the sensing rate in between these bounds depending on the network state while supporting the application-level QoS in terms of the detection delay.

Let us suppose there is L different levels of the sensing rate, c_0, c_1, ..., c_{L-1}, with c_{L-1} being the highest. Let c_c be the current value for the sensing rate level. Whenever a STEP update occurs at a node, the node computes FEDP or EToE and accordingly a new sensing rate level c_m is determined. If $c_m \neq c_c$, then the node changes the sensing rate to c_m. The way that a camera adapts its sensing rate is obviously similar to the way that a radio does, yet we want to point out that the duty cycles of radios and cameras are not necessarily the same, because the minimum and maximum duty cycles of them are controlled by different requirements with different thresholds.

Note that there could be a delay between the time when a node detects a new object within its sensing range and the time when the node reacts to the new detection in terms of communication activity (such as cluster formation and joining the existing cluster) since the radio of the node could be in the sleep mode due to duty cycling. Even though the node is already a cluster member, there is still a delay to report the measurement to the cluster head since the cluster head could be in the sleep mode.

To minimize such delay, we can schedule an image sensing in such a way that the radio is expected to start its active period immediately after the entire image processing chain is completed from the image acquisition to the high-level processing so that the radio turns on with the most up-to-date results. Figure 6.3(a) shows how such scheduling can take place along with radio duty cycling.

Even when the radio is sleeping and not scheduled to be turned on after the completion of all the image processing, a node with reactive approaches, however, still needs to capture images periodically at a high rate so as to trigger the radio to be turned on in case the radio is off when an object is detected. As mentioned earlier, the sensing rate at nodes with reactive approaches is determined based on the node-level detection delay. Provided that a node can make predictions on the future state of the object using the event tracker in our framework, however, it is unnecessary to capture images unless the radio is scheduled to be on after the completion of the image processing. Thus, it is possible that a node controls its sensing intervals according to the schedule of radio activation as illustrated in Figure 6.3(b). The PSRA is therefore a natural extension of PDCA, which will be described in details in the following section.

7. PERFORMANCE EVALUATION

In this chapter, we present a series of evaluations of the proposed predictive network adaptation by tracking (PNAT) framework where it is applied to a single layer (i.e., MAC layer) and multiple layers simultaneously (i.e., MAC and application layers). As stated in the previous chapter, the predictive adaptation in the MAC layer results in predictive duty cycle adaptation (PDCA) while the predictive adaptation in the application layer the predictive sensing rate adaptation (PSRA). We first evaluate the MAC protocols with and without PDCA to show how well the dynamic change of the environment in terms of the event-driven traffic can be handled by the PDCA-enabled MAC protocol and at the same time how much energy saving can be achieved when the PNAT is applied to the MAC layer. Then we evaluate how much energy saving can be further made by applying the PNAT to both MAC and application layers (i.e., PDCA + PSRA) simultaneously compared to when the PNAT is applied only to the MAC layer (i.e., PDCA only).

We perform such evaluations based on a real wireless camera network testbed deployed in a lab environment, which will be described in detail in the following sections, as well as large-scale simulations.

7.1 Predictive Adaptation in the MAC Layer

We now present two evaluations of the PDCA-enabled MAC protocol approach with others that do not have PDCA: 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. [64], 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. [65] 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 7.1.3) 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 [6] 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.

7.1.1 Quality-of-Service Metric

Eariler in Chapter 2.1, we have shown how a typical cluster-based collaborative processing is modeled as a state transition diagram as described in Figure 2.2. We also use this state transition diagram to define the following QoS metric: *Time-Bounded Parameter Estimation Accuracy* (TIBPEA) [17]. As to the parameter that should become the focus of this accuracy, we leave that to the user of this metric. The choice of the parameter would depend on what a WCN is being used for. If suppose a WCN is being used for tracking targets, then the accuracy achieved would concern target localization assuming that it is moving at a certain speed and that a node cluster (as it is propagating with the target) has only limited time to make inferences about the target.

While TIBPEA applies straightforwardly at a high-level in the manner explained, it is possible to create a purely communication version of this metric by defining it as *the rate of successful internode message exchange within a specified time period.* Obviously, the greater the reliability with which the cluster members can communicate with each other, the greater the accuracy of any parameter that must be computed collaboratively. When defined in this manner, TIBPEA is computed by the average percentage of neighbors that successfully reply to the broadcast messages in State 3 of the state transition diagram within a certain timeout period. In the context of visual processing in WCNs, TIBPEA can be interpreted as the precision with which a vision task can be completed by a node cluster in a time-bounded manner in the presence of bursty communications entailed by collaborative computing amongst the cluster members.

Consider again the WCN shown in Figure 4.1. Suppose an event occurs to 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 5.4 (a). Then, when Node B transmits a 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 5.4 (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 in performance evaluation with other approaches.

Obviously, the new QoS evaluation measure we have introduced, TIBPEA, is application specific. But, we believe, that that is the way it should be. It would be much too naive to assume that a WSN designed for keeping track of suspects in a crowded marketplace would work equally well for keeping track of high-speed traffic at a busy interchange. That is, the evaluation of a WSN must be specific to a category of applications and the research community must specify a suite of vision tasks for measuring the QoS for each category. For our research, we have chosen to use TIBPEA for evaluating the proposed PDCA method in a WCN for tracking simple objects.

7.1.2 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*.

Table 7.1: Summary of simulation parameters.

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	$2400Sec.$

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 [66] 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.

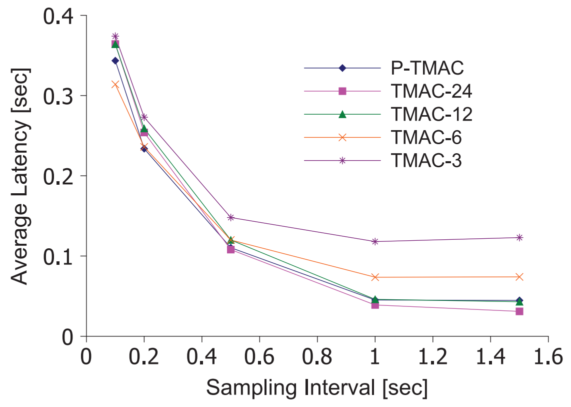
We compare the performance between P-TMAC and TMAC. The base frame length of P-TMAC is set to $T = 1000ms$, its active period to $30ms$, and its frame length is allowed to vary among $N = 4$ levels, corresponding to T, $T/2$, $T/4$, and $T/8$, that is, $M = 2$. Since the active period remains constant, 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 7.1.

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 collaborative 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.

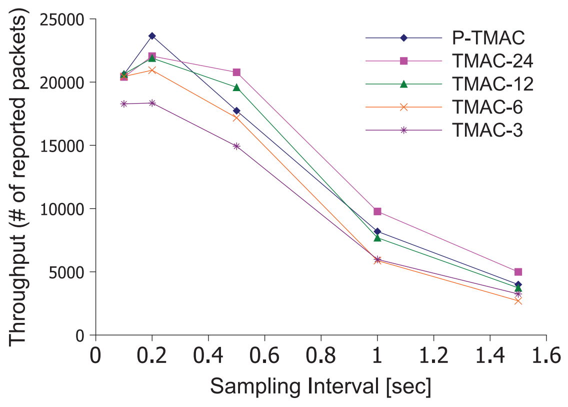
7.1.2.1 Individual Processing and Reporting Scenario

Because of the adaptive frame length design, P-TMAC is expected to show performances in between TMAC-3 and TMAC-24. Figure 7.1(a) shows that the latency of P-TMAC is comparable to that of TMAC-24 at different sampling intervals. Figure 7.1(b) shows the throughput evaluation results. Obviously, shorter sampling intervals entail higher packet rates.

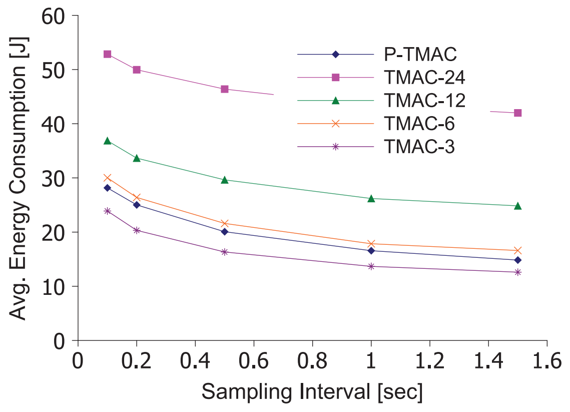
To interpret this result, let us define the period from the time an object of interest enters the sensing field of a node to the time it leaves it as the *sensing round*. Let us also define the first packet transmitted during each sensing round as the *link initializing packet*. With the same object motion, higher sampling rate causes more packet generation per sensing round, resulting in a small proportion of link initializing packets 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 7.1(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.



(a)

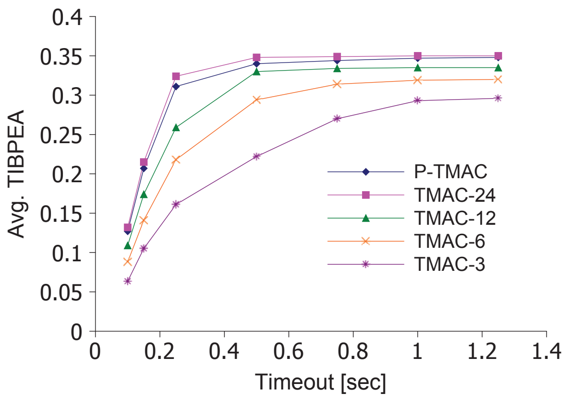


(b)

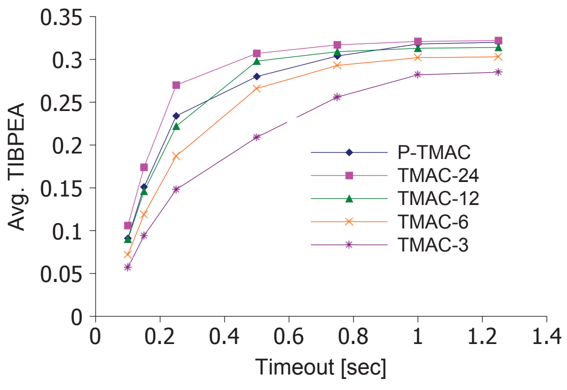


(c)

Figure 7.1.: 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.



(a)



(b)

Figure 7.2.: Simulation results of the average TIBPEA with different average target speeds: (a) 6m/s and (b) 24m/s.

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 7.1 (c). It implies that P-TMAC substantially improves the tradeoff between energy and latency compared to TMAC.

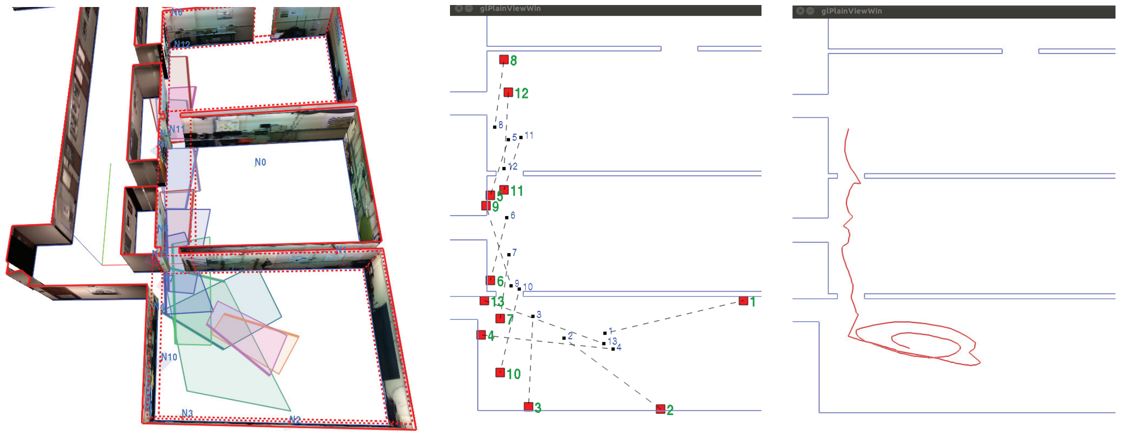
7.1.2.2 Collaborative Processing and Reporting Scenario

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 7.2. 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.

7.1.3 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 7.3(a) and the plan view in Figure 7.3(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 7.3(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 7.3(b) 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 [67], we will not go into them here. Figure 7.3(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



(a) (b) (c)

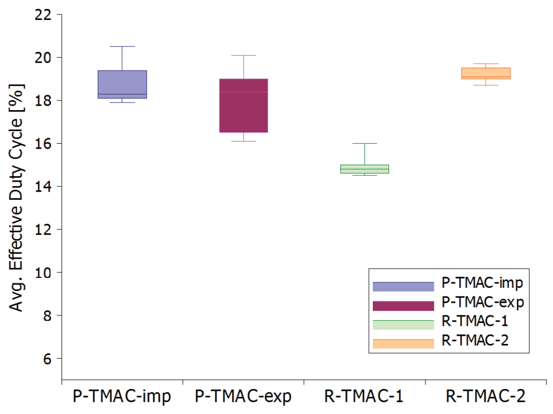
Figure 7.3.: The RVL wireless camera network testbed used for this validation consists of 13 Imote2 motes with cameras deployed across three rooms: (a) A 3D model of the testbed with the sensing region of each camera depicted as a colored polygon on the floor; (b) A plan view of the testbed with the 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.

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.

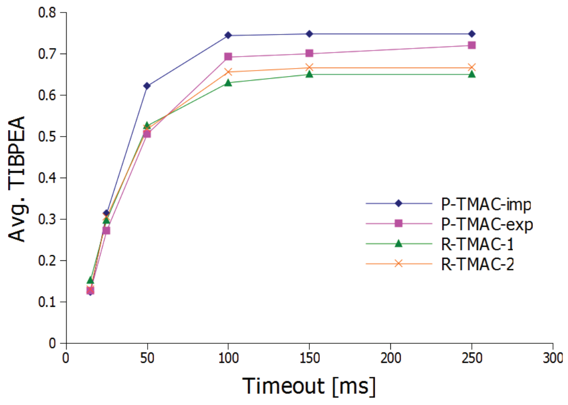
With regard to evaluating the effectiveness of the duty cycling achieved, we use the TIBPEA QoS metric and energy efficiency. Our choice of TIBPEA is dictated by the fact that the more traditional network-level performance metrics such as end-to-end latency and throughput do not capture the performance 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 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

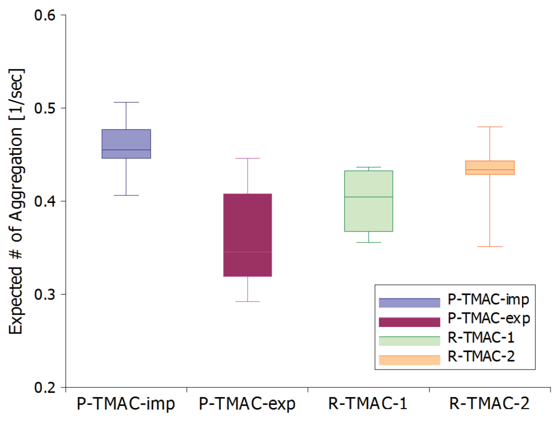


(a)

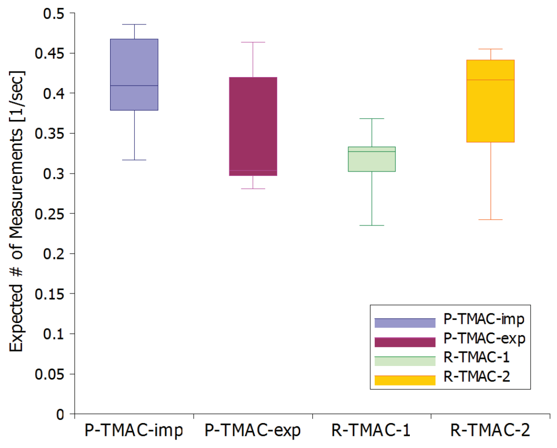


(b)

Figure 7.4.: Performance comparison in terms of (a) energy efficiency measured as the average effective duty cycle and (b) the average TIBPEA with varying timeouts.



(a)



(b)

Figure 7.5.: 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.

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 7.4 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 7.4(a), and *the average TIBPEA,* as shown in Figure 7.4(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 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 7.4(b) shows that P-TMAC-imp outperforms the other three approaches in terms of the average TIBPEA. Figure 7.4(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 7.4(b), on the basis of the average TIBPEA results shown in Figure 7.4(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 7.4(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

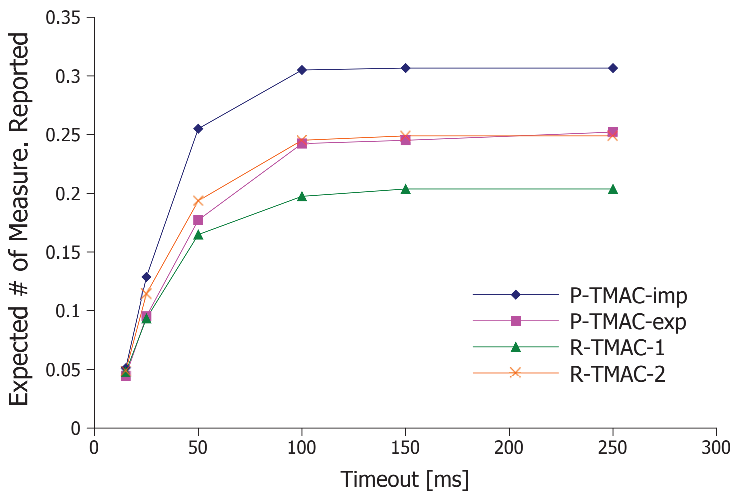


Figure 7.6.: Performance comparisons in terms of the number of measurements that a cluster head expects to receive from its cluster members within a varying timeout during object tracking. The data shown is averaged over all the clusters formed during tracking.

standpoint of the efficiency with which the clustering operations can be carried out. Average TIBPEA does not measure those effects in a network.[1]

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 7.5(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 7.5(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 7.5(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 7.6 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 7.6 strongly demonstrate how well our predictive MAC protocol supports the distributed object tracking application. In both Figures 7.5(a) and (b), it is evident that our proposed P-TMAC-imp

[1]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.

outperforms R-TMAC-1 while being comparable to R-TMAC-2 which incurs a higher energy cost. In Figure 7.6, 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.

7.2 Predictive Adaptation in Both MAC and Application Layers

We now present the performance evaluations of different network adaptation approaches such as (1) two reactive adaptation approaches and (2) a predictive adaptation approach for radio duty cycling in the MAC layer (i.e., RDCA and PDCA, respectively) and (3) a predictive adaptation of both the radio duty cycle in the MAC layer and the camera sensing rate in the application layer (i.e., PDCA + PSRA). The reactive adaptation method allows nodes to adapt their parameter only after detecting an event of interest. Thus, the radio duty cycle at a node is also increased only after the node itself detects an object.

Our evaluation is based on performance metrics that reflects application-level QoS for WCNs as well as energy efficiency. The QoS is in general defined in different ways in different applications. The application-level QoS metrics that we are interested in are, however, characterized by performance in clustering operations, since our primary application space is target tracking applications using a WCN, and a cluster is the one who keeps track of the state of the object while dynamically assigning different roles to different nodes and allowing nodes to join and leave the cluster as an object moves. Therefore, the performance metrics for clustering operations are intended to evaluate how well the data aggregation is and how smoothly the dynamic clustering operations are carried out within the cluster. In an ideal case, as long as a mobile object is present within the sensing coverage of the network, at least a cluster is formed with the nodes in the vicinity of the object, and the cluster is dynamically propagated without discontinuity (i.e., persistently) as the object moves. In addition, the aggregation of the measurements from the cluster members at the cluster head must be carried out for collaborative vision processing as scheduled on a regular basis with at least a certain success rate. In addition, the lifetime of the network also needs to be maximized for extended period of operation. More details on how well these requirements are met by each approach will be presented in the following sections.

To evaluate the performance of the three approaches, we chose a well-known synchronous MAC protocol known as TMAC [6] as the basic MAC protocol for our experiments. On top of TMAC, we apply each of the three approaches, resulting in synchronous MAC protocols with different adaptation strategies of network parameters such as radio duty cycle and camera sensing rate. We refer to the TMAC protocol with (1) a reactive duty cycle adaptation (RDCA) with duty cycles same as in the predictive approaches and (2) TMAC with a RDCA with a higher minimum duty cycle as as *R-TMAC-1* and R-TMAC-2, respectively, where 'R' stands for "reactive". The TMAC protocols with a predictive duty cycle adaptation scheme and TMAC also with a predictive sensing rate adaptation as well as PDCA are referred to as *P-TMAC-single* (i.e., TMAC with PDCA only) *P-TMAC-multi* (i.e., TMAC with both PDCA and PSRA), respectively, where 'single' and 'multi' stand for single- and multi-layer, respectively.

The reason why we also compare our proposed approach with R-TMAC-2 that has a higher minimum duty cycle than others is because R-TMAC-1 being expected to be more energy-efficient than predictive approaches due to its passiveness, we want to see the impact of spending extra energy in reactive approaches on the performance. If the conversion of energy into performance occurs, then although predictive approaches would outperform reactive ones in terms of application-level performance, it should entail the cost of higher energy expenditure and then we cannot claim that our proposed approach improves the tradeoff between energy and performance. It will turn out that due to the inherent limitation of the reactive approaches, such conversion never happen using reactive approaches even with spending extra energy.

We want to note that TMAC is already able to adapt the length of the active periods depending on the current traffic yet with fixed frame length and only in a reactive manner. Being equipped with PDCA, P-TMAC can adapt both the length of the active period and the frame length itself in a predictive manner before any event happens.

7.2.1 Evaluation Environment

In this section, we introduce the Purdue RVL Wireless Camera Network Testbed and the cluster-based object tracking application that runs on each node of the testbed.

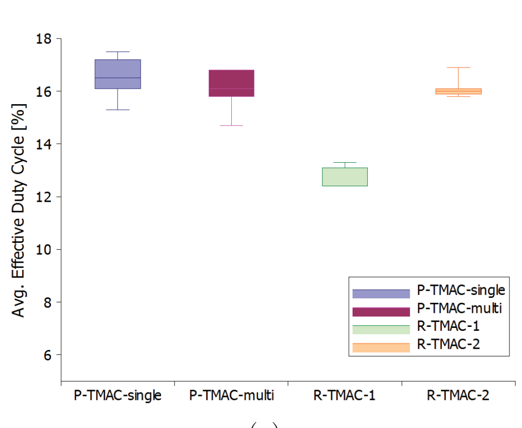
The RVL testbed consists of 13 Imote2 motes each of which is equipped with a IMB400 multimedia board [68] that includes a camera based on OV7670 image sensor. These camera nodes are deployed in a way that they cover three consecutive rooms connected through a doorway where the size of each room is roughly $20ft \times 20ft$ as shown in the 3D model of the rooms and its plan view in Figure 6.4(a) and (b), respectively. The exact sensing coverage of the testbed is also illustrated in Figure 6.4(a) as a colored polygon on the floor for each camera. The cameras are carefully deployed in such a way that most of each particular point within the sensing coverage of the network is covered by at least two cameras at the same time. Thus, as long as an object is moving within this region, the testbed is able to create a dynamic cluster with multiple cameras and track the object. Figure 6.4(c) shows an example of the track of an mobile object estimated by the testbed.

The softwares for image processing and network protocols are all implemented using TinyOS 2.x. But, obviously there are other great alternative operating systems for wireless sensor networks including Contiki [69]. The reason why TinyOS is used in this paper is only for the sake of the continued project development though the collaboration of multiple contributors in our group.

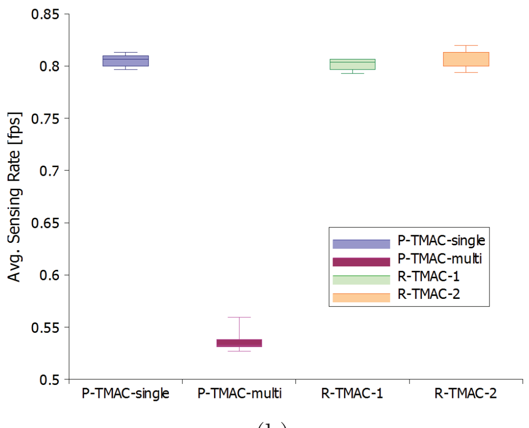
Image Processing Chain From the acquisition of an image, a visual measurement is processed by a series of image/vision processing modules in the following order; upon the reception of a request message from the application, the camera driver in TinyOS issues a command to OV7670 image sensor to capture an image. Then a frame buffer that contains an array with two bytes-long data is returned, where the two bytes are formated to include 5-, 6-, and 5-bits long red, green, and blue color information, respectively. Thus, the two bytes-long data undergo a loop to be converted into three-dimensional RGB array. A color-histogram based blob detector takes this image as an input and computes the center of mass of the detected blobs by recursively finding connected points. Note that each recursion to find connected points among the neighboring pixels increases the stack size. Due to the limited amount of memory, the maximum depth of the stack is also limited. Thus, the center of mass of a large blob cannot be correctly computed at once by a single run of the recursion-based blob detector, resulting in multiple segments for a single large blob. A connected component labeling algorithm is thus required to find the actual center of mass of the blob. The global coordinate of the center of mass of the detected blobs and their sizes are finally obtained and used later as a measurement for tracking purpose.

The processing time for the entire image processing chain from the acquisition to the blob detection takes roughly $900ms$ on average with the camera node configured that the microprocessor of each Imote2 mote runs at $208MHz$ and the image format is the color image with size of 320×240.

Note that the camera driver supported in TinyOS is not proprietary and thus not optimized. The camera driver provided in TinyOS is rather extremely primitive in that image acquisition and processing is not pipelined so that only after the image processing is completed, a new image acquisition process can start. Although the OV7670 camera supports image capturing rate up to $30fps$, therefore, due to the primitive camera driver support and limited communication between Imote2 node and the IMB400 multimedia board where the camera is located, the achievable maximum



(a)



(b)

Figure 7.7.: Performance comparisons in terms of (1) the average effective duty cycle of radio and (2) the average sensing rate of camera. Lower indicates better energy efficiency for both cases.

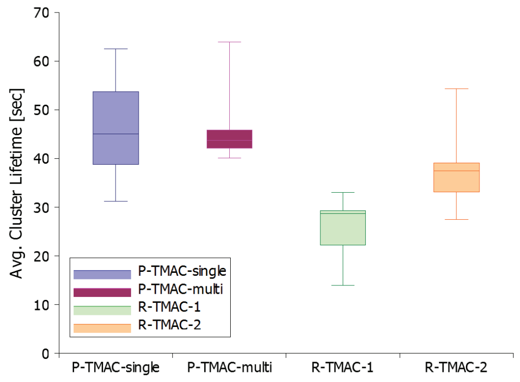
camera sensing rate with minimal image processing is roughly less than $2fps$. There is, of course, a huge room to optimize the software further for more efficient processing and capturing of images, yet we leave it as a future work and consider it as a given condition for this performance evaluation.

Dynamic Cluster-based Object Tracking Each node runs a Kalman filter-based distributed object tracking application on top of a clustering protocol as presented in [12]. When a new object is detected by a node using the aforementioned vision processing, the node broadcasts a message to see if there's any existing cluster for this object. If so, the node joins the existing cluster as a cluster member and start contributing to the data aggregation for collaborative object tracking. If not, on the other hand, the node declares itself as a cluster head and starts tracking while allowing other nodes to join if they are detecting the same object. If the object is out of sight of a cluster member, then the cluster member sends a message saying that it is leaving the cluster. If a cluster head is no longer detecting the object, then it will also broadcast a message that triggers a procedure among the cluster members to elect a new cluster head.

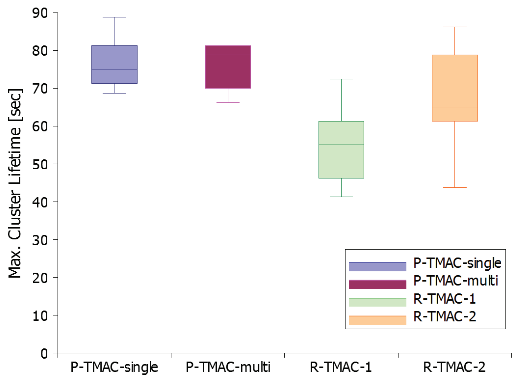
For each round of data aggregation, the cluster head sets a timeout during which the cluster members are allowed to report their measurements. Upon the expiration of the time out, the cluster head estimates the current location of the object using the Kalman filter employed. The final estimate is reported to the base station, and the cluster starts a next round of aggregation. The measurements arrived at the cluster head after the timeout expires are ignored since new measurements will be available for the next round of aggregation.

Network Parameter Configuration For the adaptive duty cycling of radio in our experiments, the exponentially varying frame length is set to be doubled or halved in case that a change of duty cycle occurs. Each node is configured to choose three different levels of duty cycle, thus the maximum duty cycle is four times higher than the minimum duty cycle. The length of an active period of a frame is set to $300ms$ while the base frame length is $4800ms$. This translates into the maximum duty cycle being $300/(4800/2^2) = 25\%$ and the minimum duty cycle being $300/(4800/2^0) = 6.25\%$. With regard to the object tracked in these experiments, we used a toy vehicle that can be navigated with a handheld controller.

The sensing schedule of the camera at a node with PSRA scheme can be matched to the radio sleep and wake-up schedule in such a way that the end of the image processing would become the beginning of the active period of radio as said earlier. Since the frame length of radio at a node is one of $4800ms$, $2400ms$ and $1200ms$, the nodes with PSRA are also set to capture an image at every $4800ms$, $2400ms$ or $1200ms$, depending on the state of the object of interest relative to the field-of-view of the nodes, resulting in the sensing rate being in between $0.83fps$ and $0.21fps$. The nodes without PSRA, on the other hand, are set to capture an image at every $1200ms$, that is, at $0.83fps$.



(a)



(b)

(c)

Figure 7.8.: Performance comparisons in terms of (a) the average and (b) the maximum lifetime of a cluster that indicate how far a cluster can be smoothly propagated while tracking a mobile object.

7.2.2 Evaluation on Clustering Operations

If there is an object of interest within the sensing coverage of the network, then a cluster with the nodes that are detecting the same object will be formed in the vicinity of the object. Since the cluster is the one who carries out the object tracking task, in this section, we therefore evaluate how well the clustering operations take place with support from different network adaptation strategies.

Since the cluster is supposed to monitor the object during its presence, the cluster must be dynamically propagated in a timely manner as the object moves. To keep track of the object without discontinuity, therefore, how long the dynamic cluster can follow the object is a key metric to evaluate how well the underlying network protocol stack supports the application. Since the discontinuity in cluster propagation directly indicates the tracking failure, we measure how persistent the cluster-based object tracking is carried out in terms of *the average and maximum lifetime of a cluster*.

Once a cluster head loses the object, then the dynamic clustering protocol [12, 16] that is currently employed for this evaluation lets the cluster head broadcast a message that would trigger the cluster head re-election process and goes to an idle mode, hoping that some of the neighboring nodes that are detecting the same object receive the message. Unless at least a node among the nodes that are detecting the same object is in active mode when the message was broadcasted by the former cluster head, the cluster head re-election process would not take place, resulting in forming a new cluster for the same object yet with the initialization of the state of the object since the previous cluster has been dismissed and thus there is no clue on whether the object is previously seen or not. As shown in Figure 7.8(a) and (b), such failure of cluster propagation occurs more frequently in the reactive approach, R-TMAC-1, than the predictive approaches. Providing more energy and thus more bandwidth to the reactive approach, which results in R-TMAC-2, increase the performance in terms of the persistent tracking yet does not exceed the other predictive approaches. Recalling that P-TMAC-multi allows nodes to have a lower sensing rate for energy conservation while P-TMAC-single keeps nodes to have a fixed sensing rate, it is clearly shown in the figure that P-TMAC-multi achieves comparable performance in terms of persistent clustering while providing higher energy efficiency than P-TMAC-single as will be shown later.

In addition to the persistency in tracking an object, the number of measurements that contributes to a single aggregation is also a important metric in tracking performance since that significantly affects the degree of the uncertainty of the estimated object state. Thus, we also measure *the average number of cluster members in a cluster*, *the expected rate of measurements generated at the cluster members* that can potentially be reported to the cluster head, and *the expected rate of measurements that are actually received at the cluster head within a varying timeout*. In an ideal case where there is no packet loss, the later two metrics must yield the same results. In practice, however, a significant amount of packet loss or an excessive delay could occur when the measurements are reported from the cluster members to the cluster head due to severe contention for the medium — the highest peak of traffic in fact occurs in this data aggregation stage. Figure 7.10 empirically shows how much measurements can be successfully reported from cluster members to the cluster head within a certain timeout during tracking. Given a timeout, there could be up to $40\% - 60\%$ of packet loss since the measurements received after the timeout expires would be considered to be useless. Even with a large timeout, the packet loss rate could reach easily around 20% for all of the approaches.

In Figure 7.9(a) and (b), the advantages of employing the predictive approaches over the reactive ones in terms of the metrics for the data aggregation seem marginal yet it is more evident in Figure 7.9(c) that shows how much measurements are expected to arrive at the cluster head within a certain timeout while tracking a mobile object. Given the fact that the RVL testbed consists of only 13 Imote2-based wireless cameras, which is honestly a small-scale testbed, we admit that the testbed may not be sufficiently large to empirically demonstrate the performance evaluation, yet we

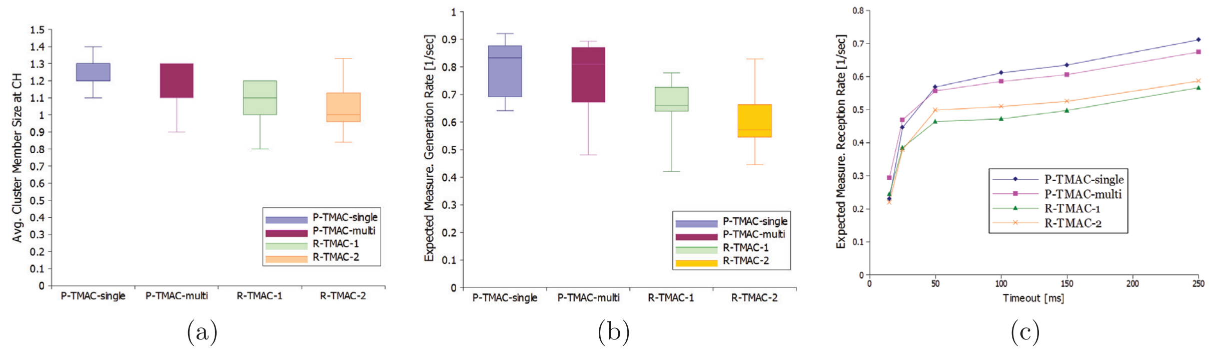


Figure 7.9.: Performance comparisons in terms of (a) the average number of cluster members in a cluster; (b) the expected rate of measurement generation at the cluster members that can potentially be reported to the cluster heads during tracking; and (c) the expected rate of measurements that are actually delivered to the cluster heads from the cluster members within a varying timeout for the data aggregation using the Kalman filter. The data shown is averaged over all the clusters formed during tracking.

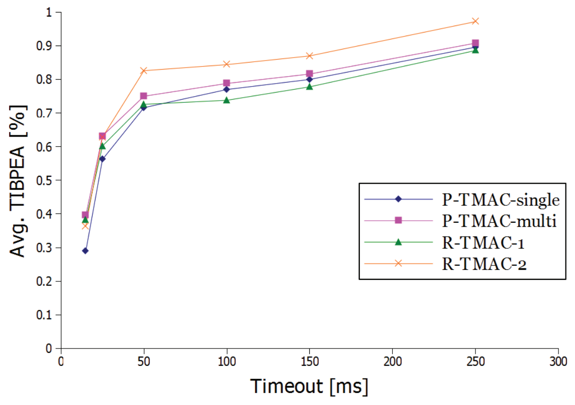


Figure 7.10.: Performance comparisons in terms of the time-bounded parameter estimation accuracy (TIBPEA) [17] that is computed by the average percentage of the cluster members that successfully reply to the cluster head within a certain timeout period.

believe the advantages of our proposed approach would be even more evident if a larger testbed is used.

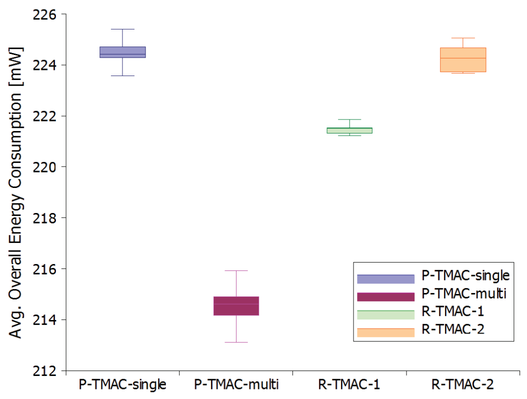


Figure 7.11.: Performance comparisons in terms of the average overall energy consumption for communication and image processing by radio, camera, and microprocessor in different adaptation approaches on Imote2-based wireless camera platforms.

7.2.3 Evaluation on Energy Efficiency

There is no doubt that the energy efficiency is of utmost importance. To demonstrate and compare the energy efficiency achieved by different network adaptation strategies, we measure *the average effective duty cycle* for the radios and *the average sensing rate* for the cameras. The average effective duty cycle reflects how efficiently the radio sleep/wakeup scheduling is controlled and is computed by the total active duration of the radios divided by the total running time.

As expected, Figure 7.7(a) shows that the reactive approach with the same duty cycling configuration with the predictive approaches, which is R-TMAC-1, consumes less energy in radio than others; however, our proposed approach that can also adapt the duty cycle of the camera as well as the radio, which is P-TMAC-multi, conserves more energy in camera with a much lower sensing rate on average as shown in Figure 7.7(b).

To assess the overall energy saving in both radio and camera, we establish an energy model as the following:

$$\begin{aligned} E &= D_r \cdot P_{ron} + (1 - D_r) \cdot P_{roff} \\ &\quad + D_c \cdot P_{con} + (1 - D_c) \cdot P_{coff} \\ &\quad + D_p \cdot P_{pon} + (1 - D_p) \cdot P_{poff}, \end{aligned}$$

where D_r, D_c, and D_p indicate the duty cycle of radio ('r'), camera ('c'), and microprocessor ('p'), respectively, and P_{ron}, P_{roff}, P_{con}, P_{coff}, P_{pon}, and P_{poff} indicate the power consumption when each of radio, camera, and microprocessor is at on and off state, respectively. The duty cycle of camera and microprocessor can be computed as the rate of the time for acquiring an image and for processing the image from low- to high-level, respectively, using the information in Table 6.2. The average overall overhead in terms of energy consumption for communication and image processing by radio, camera, and microprocessor in Imote2-based wireless camera platform is shown in Figure 7.11 where it clearly demonstrates that our proposed simultaneous multi-layer adaptation of the duty cycles of both radio and camera, which is referred to as P-TMAC-multi, significantly outperforms the single-layer and/or reactive adaptation approaches in terms of the energy efficiency while providing the application-level performance in terms of clustering operations and tracking accuracy, which is higher than reactive approaches and comparable to the state-of-the-art predictive approach, P-TMAC-single.

8. CONCLUSION AND FUTURE WORK

In this dissertation, we first investigated unique characteristics of Wireless Camera Networks (WCNs) in terms of communication, resource demand and quality-of-service and accordingly necessary design considerations for such networks. We then showed existing adaptive approaches including adaptive MAC protocols are not suitable for WCNs. As a solution, we proposed the Predictive Network Adaptation by Tracking (PNAT) framework that actively adapts the network parameters of nodes in advance before the appearance of an object of interest and the abrupt change in traffic pattern triggered by the object detection. To enable this ability, we introduced an concept of tracking an object through indirect sensing in the MAC layer. By localizing the current object beyond the direct sensing range and predicting the future state of the object, each node can determine and adapt to the proper level of parameters within a range prior to any significant event in order to handle the upcoming traffic in a timely and efficient manner or promptly detect an object. The realization of PNAT in the MAC and application layers resulted in the predictive duty cycle adaptation (PDCA) and the predictive sensing rate adaptation (PSRA), respectively. Since the PDCA scheme allows each node to have a different duty cycle based on its local decision, we also proposed an efficient algorithm that enables successful communication among nodes with different duty cycles. As a consequence, the PNAT framework improves the fundamental tradeoff between energy efficiency and application-level performance by supporting the QoS of the applications while minimizing the energy consumption.

The performance evaluations on the RVL wireless camera network testbed with 13 real Imote2-based wireless cameras and on a large-scale simulation demonstrated that (1) the TMAC with the PDCA scheme outperforms the original TMAC in terms of such network performance metrics, (2) our predictive framework outperforms reactive approaches, and (3) the simultaneous multi-layer network adaptation does so the single layer-based adaptation.

To the best of our knowledge, the PNAT scheme is (1) the first attempt to adapting the duty cycle of nodes by tracking the object of interest in the MAC layer, (2) the first attempt to employing a Kalman filter in the MAC layer for that purpose, (3) the first attempt to solving the problem of efficiently communicating among nodes with different duty cycles in adaptive synchronous MAC protocols, and (4) the first attempt to adapting the parameters in the MAC and application layers simultaneously.

One fundamental issue associated with any adaptive approach is how to decide the range of adaptation, that is, what should be the optimal lower and upper bounds within which a network parameter such as duty cycle is adapted. The existing adaptive MAC approaches as well as our proposed approach focus only on "how to" adapt the duty cycle assuming the range of adaptation is given. For optimal adaptation, however, the range "within which" a node adapts its duty cycle should be addressed because the duty cycle needs to be adapted within the optimal bounds in an optimal way. Since this report proposed an "optimal way" of adapting duty cycle for WCNs, the next step would include to obtain the optimal range of duty cycle adaptation for the same networks.

A further step toward the extension of the PNAT framework is to allow nodes to perform a predictive network adaptation using PNAT framework even in the presence of multiple objects within the augmented sensing field since the PNAT framework presents the case of single object of interest.

LIST OF REFERENCES

LIST OF REFERENCES

[1] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC: a hybrid MAC for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, 2008.

[2] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "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, (New York, NY, USA), pp. 307–320, ACM, 2006.

[3] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 95–107, ACM, 2004.

[4] A. El-Hoiydi and J. Decotignie, "WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks," *Lecture Notes in Computer Science*, vol. 3121, pp. 18–31, 2004.

[5] W. Ye, J. Heidemann, and D. Estrin, "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, 2002.

[6] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 171–180, ACM New York, NY, USA, 2003.

[7] Y. Nam, H. Lee, H. Jung, T. Kwon, and Y. Choi, "An adaptive MAC (A-MAC) protocol guaranteeing network lifetime for wireless sensor networks," in *Proceedings of 12th European Wireless Conference (EW 2006), Athens, Greece*, 2006.

[8] P. Lin, C. Qiao, and X. Wang, "Medium access control with a dynamic duty cycle for sensor networks," in *2004 IEEE Wireless Communications and Networking Conference, 2004. WCNC*, vol. 3, 2004.

[9] S. Liu, K.-W. Fan, and P. Sinha, "CMAC: An energy-efficient mac layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 4, pp. 1–34, 2009.

[10] M. Healy, T. Newe, and E. Lewis, "Wireless sensor node hardware: A review," in *Sensors, 2008 IEEE*, pp. 621–624, 2008.

[11] O. Dousse, C. Tavoularis, and P. Thiran, "Delay of intrusion detection in wireless sensor networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, (New York, NY, USA), pp. 155–165, ACM, 2006.

[12] H. Medeiros, J. Park, and A. C. Kak, "Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 448–463, 2008.

[13] J. Park, P. C. Bhat, and A. C. Kak, "A look-up table based approach for solving the camera selection problem in large camera networks," in *in Proceedings of the International Workshop on Distributed Smart Cameras*, 2006.

[14] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, 2009.

[15] Y. Charfi, N. Wakamiya, and M. Murata, "Challenging issues in visual sensor networks," *Wireless Communications, IEEE*, vol. 16, no. 2, pp. 44 –49, 2009.

[16] H. Medeiros, J. Park, and A. C. Kak, "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*, pp. 203–210, Sept. 2007.

[17] P. J. Shin, J. Park, and A. C. Kak, "A QoS Evaluation Testbed for MAC Protocols for Wireless Camera Networks," in *First ACM/IEEE International Conference on Distributed Smart Cameras, 2007. ICDSC '07*, pp. 235–242, Sept. 2007.

[18] G. Mainland, D. C. Parkes, and M. Welsh, "Decentralized, adaptive resource allocation for sensor networks," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, (Berkeley, CA, USA), pp. 315–328, USENIX Association, 2005.

[19] J. Kho, A. Rogers, and N. R. Jennings, "Decentralized control of adaptive sampling in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, pp. 19:1–19:35, June 2009.

[20] D. Pepyne, D. Westbrook, B. Philips, E. Lyons, M. Zink, and J. Kurose, "Distributed collaborative adaptive sensor networks for remote sensing applications," in *American Control Conference, 2008*, pp. 4167–4172, 2008.

[21] P. Hurni and T. Braun, "MaxMAC: A Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks," in *Wireless Sensor Networks*, vol. 5970 of *Lecture Notes in Computer Science*, pp. 289–305, Springer Berlin Heidelberg, 2010.

[22] M. Anwander, G. Wagenknecht, T. Braun, and K. Dolfus, "BEAM: A Burst-aware Energy-efficient Adaptive MAC protocol for Wireless Sensor Networks," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pp. 195 –202, 2010.

[23] R. Jurdak, P. Baldi, and C. V. Lopes, "Adaptive low power listening for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 988–1004, 2007.

[24] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, "ptunes: runtime parameter adaptation for low-power mac protocols," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, (New York, NY, USA), pp. 173–184, ACM, 2012.

[25] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: application driver for wireless communications technology," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 20–41, April 2001.

[26] A. Farina, G. Golino, A. Capponi, and C. Pilotto, “Surveillance by means of a random sensor network: a heterogeneous sensor approach,” in *Information Fusion, 2005 8th International Conference on*, vol. 2, p. 8 pp., 2005.

[27] G. Mainland, D. C. Parkes, and M. Welsh, “Decentralized, adaptive resource allocation for sensor networks,” in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI’05, (Berkeley, CA, USA), pp. 315–328, USENIX Association, 2005.

[28] S.-H. Choi, D. Perry, and S. Nettles, “A software architecture for cross-layer wireless network adaptations,” in *Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on*, pp. 281–284, 2008.

[29] O. Karaca and R. Sokullu, “Comparative study of cross layer frameworks for wireless sensor networks,” in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pp. 896–900, 2009.

[30] R. Kuntz, A. Gallais, and T. Noel, “Medium access control facing the reality of wsn deployments,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 22–27, June 2009.

[31] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, “MAC Essentials for Wireless Sensor Networks,” *Communications Surveys Tutorials, IEEE*, vol. 12, no. 2, pp. 222 –248, 2010.

[32] K. K. II and P. Mohapatra, “Medium access control in wireless sensor networks,” *Computer Networks*, vol. 51, no. 4, pp. 961 – 994, 2007.

[33] J. Jubin and J. Tornow, “The DARPA packet radio network protocols,” *IEEE Proceedings*, vol. 75, pp. 21–32, Jan. 1987.

[34] G. Halkes, T. Van Dam, and K. Langendoen, “Comparing energy-saving MAC protocols for wireless sensor networks,” *Mobile Networks and Applications*, vol. 10, no. 5, pp. 783–791, 2005.

[35] P. Hurni, T. Braun, B. K. Bhargava, and Y. Zhang, “Multi-hop Cross-Layer Design in Wireless Sensor Networks: A Case Study,” *Wireless and Mobile Computing, Networking and Communication, IEEE International Conference on*, vol. 0, pp. 291–296, 2008.

[36] P. Hurni and T. Braun, “An Energy-Efficient broadcasting scheme for unsynchronized wireless sensor MAC protocols,” in *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, pp. 39–46, Feb. 2010.

[37] A. Ephremides and O. Mowafi, “Analysis of a hybrid access scheme for buffered users-probabilistic time division,” *IEEE Transactions on Software Engineering*, pp. 52–61, 1982.

[38] I. Rhee, A. Warrier, J. Min, and L. Xu, “DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks,” in *MobiHoc ’06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), pp. 190–201, ACM, 2006.

[39] K. Jamieson, H. Balakrishnan, and Y. Tay, "Sift: A MAC protocol for event-driven wireless sensor networks," *Lecture Notes in Computer Science*, vol. 3868, p. 260, 2006.

[40] M. C. Vuran and I. F. Akyildiz, "Spatial correlation-based collaborative medium access control in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 316–329, April 2006.

[41] V. Namboodiri and A. Keshavarzian, "Alert: An Adaptive Low-Latency Event-Driven MAC Protocol for Wireless Sensor Networks," in *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, (Washington, DC, USA), pp. 159–170, IEEE Computer Society, 2008.

[42] Z. H. Mir, Y.-B. Ko, S. Park, and C. S. Pyo, "Ec-mac: A cross-layer communication protocol for dynamic collaboration in sensor networks," in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pp. 1 –6, 2010.

[43] C. Gui and P. Mohapatra, "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, (New York, NY, USA), pp. 129–143, ACM, 2004.

[44] N. Vasanthi and S. Annadurai, "An Adaptive Energy Efficient Low Latency Sleep Schedule for Target Tracking Sensor Networks," *IJCSNS*, vol. 8, no. 4, p. 291, 2008.

[45] V. S. Tseng and E. H.-C. Lu, "Energy-efficient real-time object tracking in multi-level sensor networks by mining and predicting movement patterns," *Journal of Systems and Software*, vol. 82, no. 4, pp. 697 – 706, 2009. Special Issue: Selected papers from the 2008 IEEE Conference on Software Engineering Education and Training (CSEET08).

[46] W. Zhang and G. Cao, "Dctc: dynamic convoy tree-based collaboration for target tracking in sensor networks," *Wireless Communications, IEEE Transactions on*, vol. 3, no. 5, pp. 1689 – 1701, 2004.

[47] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, (New York, NY, USA), pp. 70–84, ACM, 2001.

[48] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: a wireless sensor network for target detection, classification, and tracking," *Computer Networks*, vol. 46, no. 5, pp. 605 – 634, 2004. Military Communications Systems and Technologies.

[49] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 301 – 308, 2005.

[50] N. Shrivastava, R. M. U. Madhow, and S. Suri, "Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, (New York, NY, USA), pp. 251–264, ACM, 2006.

[51] Z. Wang, E. Bulut, and B. Szymanski, "A distributed cooperative target tracking with binary sensor networks," in *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pp. 306 –310, May 2008.

[52] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, (New York, NY, USA), pp. 150–161, ACM, 2003.

[53] J. Singh, U. Madhow, R. Kumar, S. Suri, and R. Cagley, "Tracking multiple targets using binary proximity sensors," in *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, (New York, NY, USA), pp. 529–538, ACM, 2007.

[54] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000. 10.1023/A:1008935410038.

[55] C. Hue, J. Le Cadre, and P. Pérez, "Tracking multiple objects with particle filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 791–812, 2002.

[56] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[57] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling, "Imote2: Serious computation at the edge," in *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International*, pp. 1118 –1123, aug. 2008.

[58] M. Coates, "Distributed particle filters for sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, (New York, NY, USA), pp. 99–107, ACM, 2004.

[59] M. Coates and G. Ing, "Sensor network particle filters: motes as particles," in *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on*, pp. 1152 –1157, 2005.

[60] G. Welch and G. Bishop, "An introduction to the Kalman filter," *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.

[61] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler, "Elapsed time on arrival; a simple and versatile primitive for canonical time synchronisation services," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 1, no. 4, pp. 239–251, 2006.

[62] B.-S. A. Jung, Deokwoo and A. Savvides, "imote2 node and enalab camera module power measurements," *ENALAB Technical Report-090601.*

[63] Imote2 Datasheet, Crossbow Technology `http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf`.

[64] G. Bianchi, L. Fratta, and M. Oliveri, "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, pp. 392 –396 vol.2, Oct. 1996.

[65] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-efficient surveillance system using wireless sensor networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, (New York, NY, USA), pp. 270–283, ACM, 2004.

[66] A. Boulis, "Castalia: revealing pitfalls in designing distributed algorithms in WSN," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 407–408, ACM, 2007.

[67] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[68] IMB400 Datasheet, Crossbow Technology `http://www.openautomation.net/uploadsproductos/imote2_imb400_preliminary.pdf`.

[69] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pp. 455–462, 2004.

VITA

VITA

Paul J. Shin is a Ph.D. candidate in the School of Electrical and Computer Engineering at Purdue University, West Lafayette. He received his B.E. in Electrical Engineering from Korea University in 2006. His research interests are on various networking aspects of delay-sensitive applications for Wireless Sensor Networks, including cross-layer optimization for real-time applications on Wireless Camera Networks, application-specific Qualty-of-Service issues, and network adaptation. He is a recipient of the best poster award of the ACM/IEEE Conference on Distributed Smart Cameras, 2008.