Information-driven Tracking and Access Control in Wireless Ad hoc and Sensor Networks

Gahng-Seop Ahn

Submitted in partial fulfillment of the
Requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2009

ABSTRACT

Information Driven Tracking and Access Control in Wireless Ad hoc and Sensor Networks

Gahng-Seop Ahn

This dissertation addresses three important problems in wireless ad hoc and sensor networks. The subject of wireless ad hoc and sensor networks is identified as one of the most important areas of research for future wireless systems. The robustness, flexibility, and adaptability of these self-organizing networks unleash unprecedented opportunities for a wide spectrum of applications. In addition, wireless sensor networks provide information around our daily lives as well as information about the physical environment at a finer granularity, yet at a larger scale than has been possible before. Wireless ad hoc and sensor networks are changing the way we perceive and share information of all kinds. This thesis addresses three important problems in wireless ad hoc and sensor networks, i.e., (1) service differentiation, (2) the funneling problem, and (3) the mobile tracking problem. To solve these problems, this thesis presents three information-driven systems, which are MetroTrack, Funneling-MAC, and SWAN. Although the three systems address three different problems in different contexts, they share a fundamental design principle, namely information-driven principle, in that the systems utilize the information of the dynamic environment that may affect the performance and robustness of the networks. Wireless ad hoc and sensor networks have highly dynamic and unpredictable characteristics because the nodes in the networks self-organize into multi-hop topologies on the fly while, at the same time, they are being influenced by the mobility of the nodes and the time-varying radio condition. Therefore, the systems must be designed to be responsive to changes in the highly-dynamic networks. The information-driven approach can fulfill this requirement. The first contribution of this thesis is to propose SWAN, a stateless wireless ad hoc network model, which supports service differentiation for real-time and best-effort traffic in a simple, robust, and responsive

manner. As a "stateless" model, the intermediate nodes in SWAN do not keep any per-flow or aggregate state information. Therefore, it is not necessary to establish, update, refresh, and remove per-flow state information. As a result, SWAN does not introduce any complex signaling and state control mechanisms that are required in "stateful" quality-of-service (QoS) approaches. SWAN does not require any QoS support from the MAC layer. SWAN assumes a best-effort MAC and performs feedback-based and information-driven control mechanisms to satisfy the bandwidth and delay requirements for real-time services. This thesis presents simulation, analysis, and experimental results that show that SWAN is capable of supporting real-time services with low and stable delays under various topology, traffic, and mobility conditions. Next, this thesis investigates the problem of funneling effect in wireless sensor networks. Wireless sensor networks exhibit a unique funneling effect that is a product of the distinctive many-to-one, hop-by-hop traffic pattern found in the networks. The funneling effect causes a significant increase in transit traffic intensity, collisions, congestion, packet losses, and energy drain. While congestion control and data aggregation techniques can help counter this problem, they cannot fully alleviate it. This thesis takes a different, but complementary, approach to solving this problem and presents the design, implementation, and evaluation of a localized, sink-oriented funneling-MAC capable of mitigating the funneling effect. The funneling-MAC is based on CSMA/CA being implemented network-wide, with a localized TDMA algorithm overlaid in the funneling region (i.e., within a small number of hops from the sink). The funneling-MAC represents a hybrid MAC approach that does not have the scalability problem associated with the network-wide deployment of TDMA. The funneling-MAC represents an information-driven approach in the sense that the sink node collects information about the traffic pattern and the intensity inside the funneling region and uses the information to dynamically compute the schedule and new depth of the funneling region. The final contribution of this thesis is to present a mobile tracking system using mobile phones carried by people as mobile sensors. Sensor-enabled mobile phones are better suited to track mobile events in urban areas than

traditional solutions such as static sensor fields that have limitations in scale, performance, and cost. There are, however, a number of challenges in building a mobile-tracking system using mobile phones. First, mobile sensors need to be tasked before sensing. Second, the mobility of people is uncontrolled. Finally, there is no guarantee that there will be a sufficient number of sensors around a target event. These challenges result in time-varying sensor coverage and disruptive event tracking. To address these challenges, this thesis introduces MetroTrack, a novel distributed tracking system that tracks mobile events using off-the-shelf mobile phones. MetroTrack presents the information-driven tasking algorithm to respond to dynamically moving targets and time-varying density of sensors around the target. The tasking algorithm is information-driven in the sense that each mobile sensor uses to make local decisions about whether to forward the task using local state information and sensor readings. MetroTrack also presents the prediction-based recovery algorithm that recovers the target when it is lost due to time-varying sensor coverage. MetroTrack recovers the lost target by tasking mobile sensors in close proximity to the lost target based on a prediction of its future location using a distributed Kalman-Consensus filter. A proof-of-concept prototype MetroTrack is implemented using Nokia N80 and N95 phones. The performance of MetroTrack, which cannot be fully analyzed in the small-scale, proof-of-concept testbed, is studied in large-scale simulations. The simulation results indicate that MetroTrack is robust in the presence of different mobility models and densities of mobile sensors.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my sincere appreciation to Professor Andrew T. Campbell for his support and guidance. As my advisor, he has provided me with insightful comments and suggestions. His hard working attitude and high expectation towards research have inspired me to become a better researcher. It was a pleasure as well as an honor to work with someone of his knowledge and experience.

I would like to thank Professor Mischa Schwartz for providing insightful comments on my research and for also serving on my dissertation committee. I would also like to thank Professor Henning Schulzrinne, Professor Dan Rubenstein, and Professor Gil Zussman for serving on my dissertation committee. I would like to thank Professor Reza Olfati-Saber for his input on my research. I would like to express my appreciation to Judith Hertog for helping me improve my English writing skills. I would like to thank Azlyn Smith for providing excellent services as the academic programs officer in Electrical Engineering Department. I would next express my gratitude to all my colleagues in Columbia University and Dartmouth College for their companionship and encouragements.

I would like to give my special thanks to my dear wife Hyun Jung Lim whose patient love enabled me to complete this work. I am also grateful to my parents-in-law for their encouragement. Many thanks to my sister and her family for their encouragement. I want to dedicate my thesis to my parents, Byung-Ha Ahn and Sun-Aee Park.

Chapter 1

Introduction

1.1 Overview

Wireless ad hoc and sensor networks have been identified as key areas of research and are anticipated to play an important role in future wireless systems. The robustness, flexibility, and adaptability of these self-organizing networks unleash unprecedented opportunities for a wide spectrum of applications. Such applications include environmental monitoring, surveillance, disaster recovery, vehicular networking, industrial robot networking, collaborative computing, campus networking, and social networking. In addition, wireless sensor networks provide the information around our daily lives as well as the information about the physical environment at a finer granularity, yet at a larger scale than has been possible before. Wireless ad hoc and sensor networks are changing the way we perceive and share the information around the world. This thesis addresses three important problems in wireless ad hoc and sensor networks, i.e., (1) service differentiation, (2) the funneling problem, and (3) the mobile tracking problem.

Wireless ad hoc networks are autonomous network systems that do not rely on a pre-established infrastructure. Instead, wireless nodes dynamically self-organize into ad hoc network topologies on the fly. Hence, wireless ad hoc networks can be easily deployed with

improved flexibility and reduced costs.

Wireless ad hoc networks have roots in the DARPA Packet Radio Networks (PRNet) [51] of the 1970's and the SURAN project [8] of the 1980's. The early stage of wireless ad hoc networks in the 1990's focused on military applications. Recently, non-military applications in commercial and industrial areas have become important.

Wireless ad hoc network research is still active today in the fields of routing, medium access control, resource management, service differentiation, power control, and security. The issues in the research include the decentralized topology and multi-hop communication of wireless ad hoc networks that cause two well-known problems in medium access control, namely the hidden terminal problem and the exposed terminal problem. In addition, the dynamic changes in network topology make every aspect of the research exceptionally challenging. Mobile ad hoc networks are a class of wireless ad hoc networks that are characterized by their emphasis on the mobility of the nodes in the networks. The topology of mobile ad hoc networks may change rapidly and unexpectedly because the nodes in the network move arbitrarily. When some nodes cannot communicate directly with each other due to their limited transmission ranges, every node in the network acts as a router and connects the nodes over multiple hops. Naturally, routing is one of the most important research issues in mobile ad hoc networks.

Wireless sensor networks are wireless networks of spatially-distributed sensor devices that collaboratively monitor the physical environment. According to [25], early wired-sensor networks emerged during the Cold War for military applications, such as the sound surveillance system (SOSUS) for submarine surveillance in the ocean and the networks of air defense radars for airborne warning and air traffic control. The wireless sensor networks of today emerged with the aid of recent advances in computing and communications. Tiny, low-cost sensor devices, which are built upon micro-electromechanical-system (MEMS) technology, low-power wireless communication, and low-power microprocessors, allow the deployment of wireless sensor networks in a wide range of applications.

Originally motivated by military applications on the battlefield, the application domain of wireless sensor networks has expanded to civilian areas such as disaster areas, industrial facilities, and natural habitat. Tiny, inexpensive sensor devices are suitable for random deployment in large numbers in inaccessible terrains, on the battlefield, and in disaster areas. The self-organization features of wireless ad hoc network technology play a key role in such rapid, random deployment. For this reason, a wireless sensor network is considered to be a kind of wireless ad hoc network. Wireless sensor networks are energy constrained in many cases. Sensor devices are expected to be battery powered, and the replacement of the batteries is neither easy nor cost effective. Extending the lifetime of these energy-constrained sensor devices has been one of the key research issues in wireless sensor networks. Researchers have focused on minimizing the energy consumption associated with wireless communication because wireless communication is one of the most energy-consuming operations. One approach to minimize the energy consumption is duty cycling, which means that sensor nodes turn off their radios when wireless communication is not necessary. Another approach is data aggregation in which the data are processed and compressed in the network so that the amount of data to be transmitted is reduced. The third approach is congestion control and collision avoidance. If a data packet is lost during multi-hop transmission due to congestion or collision, the energy consumed for transmitting the data is wasted. Furthermore, more energy may be required to retransmit the lost data. Congestion control and collision avoidance minimize such energy waste.

Recently, mobility in wireless sensor networks has become an important research issue. One of the early mobile sensor network approaches leverages mobile sinks that hover around the sensor field to collect data effectively. In recent research on medical applications, the sensors attached to patients and doctors represent mobile sensor nodes. Researchers are pushing towards people-centric mobile sensor networks that closely monitor the daily lives of people. Mobile phones are one of the best candidates for this purpose. Almost everyone carries a mobile phone, and some smart mobile phones are already equipped with multiple

sensors and capable of processing the sensor data. Leveraging mobile phones as mobile sensors is becoming a hot issue in sensor network research.

1.2 Problem Statement

This dissertation investigates three fundamental problems in wireless ad hoc networks and sensor networks. First, we investigate the problem of service differentiation supporting real-time services, such as voice and video over best-effort services such as file transfer and web traffic. Real-time services have certain bandwidth and delay requirements that are not easily satisfied in conventional congestion-controlled network systems. As illustrated in Figure 2.1, a general TCP-like congestion control algorithm ensures that the system operates around or preferably close to the congestion "cliff," which ensures maximum system throughput but at the cost of larger packet delays, which are not desirable for real-time services. For real-time services, the system should operate at the delay "knee" where the system throughput is almost the same as the cliff, but the buffers are significantly less loaded, so the delay is close to the minimum. How to ensure that the system will be operated in a way that is favorable for real-time services is an important problem and is the subject of Chapter 2.

Chapter 3 investigates the funneling problem in multi-hop wireless sensor networks. Typically, a wireless sensor network consists of many sensor nodes and a smaller number of sink nodes. These networks exhibit a unique funneling effect where events generated in the sensor field travel hop-by-hop in a many-to-one traffic pattern toward one or more sink points, as illustrated in Figure 3.1. This combination of hop-by-hop communications and centralized data collection at a sink creates a choke point on the free flow of events out of the sensor network. The funneling effect leads to increased transit traffic intensity and delay as events move closer toward the sink, resulting in significant packet collisions, congestion, and packet loss. At best, this leads to limited application fidelity measured at the sink and, at worst, congestion collapse of the sensor network. In addition, the sensors nearest to the

sink, typically within a smaller number of hops, loose a disproportionately larger number of packets and consume significantly more energy than sensors further away from the sink, hence shortening the operational lifetime of the overall network. Mitigating the funneling effect is an important problem to the sensor network community.

In Chapter 4, we identify the problem of uncontrolled mobility of mobile sensors in mobile tracking systems. The mobile sensors, such as mobile phones that are equipped with sensors, are carried by people. Since the mobility of people is uncontrolled, the mobility of the mobile sensors is also uncontrolled. Due to uncontrolled mobility, there is no guarantee that there is always sufficient density of mobile sensors around any given event of interest. Since the density changes over time, sometimes there will be a sufficient number of devices around the event to be tracked, and, at other times, there will be inadequate device density. The event-tracking process has to be designed assuming that the process of tracking will be periodically disrupted in response to time-varying density and, therefore, changing coverage conditions. Thus, a fundamental problem is how to recover a target when the system loses track of the target due to changing coverage.

1.3 Technical Barriers

Wireless ad hoc and sensor networks have distinctive characteristics and constraints that are significantly different from traditional networks. In the following sections, we discuss the important technical barriers in supporting service differentiation in wireless ad hoc networks, mitigating the funneling effect in wireless sensor networks, and achieving mobile event tracking using mobile sensors.

1.3.1 Service Differentiation in Wireless Ad hoc Networks

The topology and the traffic condition of wireless ad hoc networks can change dynamically due to node mobility and time-varying radio conditions. The dynamic nature of wireless ad hoc networks makes it challenging to ensure the bandwidth and delay requirements of real-time services. For these reasons, the widely accepted Internet standards, such as Resource Reservation Protocol (RSVP) [11] and Session Initiation Protocol (SIP) [89], are not directly applicable to wireless ad hoc networks.

INSIGNIA [61] is the first QoS signaling system specifically designed for wireless ad hoc networks. However, stateful, reservation-based QoS support approaches, including INSIGNIA, require complex signaling and state control mechanisms to establish, update, refresh, and remove per-flow state information in order to keep up with the changes in the topology and the traffic conditions. In addition, the requirement of per-flow state information raises a scalability issue in that the number of real-time flows that can be supported in the network is limited.

On the other hand, end-point admission control schemes that use end-to-end measurement-based admission control do not require that intermediate routers maintain the per-flow state. While these approaches clearly address the needs for scalability and service quality in wired networks, they do not address the important issue of mobility, which complicates the delivery of QoS in mobile networks. If such schemes were to be implemented in mobile networks, then the admission control guarantees would be continuously violated by mobility as admitted flows are rerouted in wireless networks.

Another technical barrier in supporting QoS for real-time application is associated with the design of the medium-access-control (MAC) protocol. The dynamic nature of wireless ad hoc networks makes it difficult to dynamically assign a central controller to maintain connection state and reservations. Because of this, best-effort, distributed MAC controllers are widely used in existing wireless ad hoc networks. There have been several proposals and

specifications to support service differentiation at the MAC layer using distributed control schemes, but none is widely used yet.

Clearly, there is a need for a simple service differentiation approach that does not require per-flow state information to be stored in the intermediate nodes and, at the same time, allows the end-to-end admission control to be updated according to the changes in topology and traffic condition of the wireless ad hoc network. Also, the service differentiation scheme should be able to operate even if the underlying MAC does not support service differentiation.

1.3.2 Mitigating the Funneling Effect in Wireless Sensor Networks

In order to respond to increased loads and congestion in sensor networks, researchers have proposed distributed congestion control algorithms, tiered network designs, and data aggregation techniques. However, as the literature indicates, these techniques alone cannot fully alleviate the problem because it is very difficult to effectively rate control traffic at aggregation points or sources to match the bottleneck conditions at the sink nodes. Due to the funneling effect, most packet losses in a sensor network occur within the first few hops from the sink. We argue that, by putting additional control within the first few hops from the sink, we can significantly improve communication performance and mitigate the funneling effect.

While several MAC protocols have been proposed for sensor networks, to the best of our knowledge none has addressed the funneling effect. As discussed in Chapter 3, existing contention-based (CSMA) MAC protocols in sensor networks are not capable of mitigating the funneling effect because of the large build-up of losses in nodes closer to the sink.

Several schedule-based (TDMA) MAC protocols that do better at mitigating the funneling effect have been proposed in the sensor-network literature. Schedule-based approaches achieve collision-free access and energy efficiency by assigning each node its own trans-

mitting and listening slots, allowing nodes to sleep when it is not their slot time. However, this approach has a scalability problem because the sink requires complete topology information to compute the TDMA schedule and because every node requires precise time synchronization.

The most suitable approach for potentially mitigating the funneling effect is a hybrid approach that combines the benefit of both CSMA and TDMA approaches. However, existing hybrid MAC protocols are either limited when dealing with the funneling effect or still too complex when computing TDMA schedule. Because the burden of computing the TDMA schedule is quite heavy, the literature does not recommend that the schedule be computed periodically. In Chapter 3, we discuss the argument that, if the schedule is computed only at startup and is not computed periodically, the schedule is susceptible to "schedule drift" because of time varying radio impairments.

1.3.3 Mobile Event Tracking in Urban Areas

Event tracking is an important application and active area of research in sensor networks. In the past, tracking applications (e.g., surveillance, hazard tracking, and wildlife monitoring) have driven the deployment of sensor networks across a number of disparate domains, such as battlefields, industrial facilities, and protected environmental areas. Urban sensing is an emerging area of interest that presents a new set of technical barriers. When we think about the tracking problem, traditional solutions that come to mind are based on the deployment of static sensor networks. Building sensor networks for urban environments requires careful planning and the deployment of a very large number of sensors in order to offer sufficient coverage density for event detection and tracking. Unless the network provides complete coverage, the location where the network will be deployed must be determined in advance. However, the events are unpredictable in time and space. We argue that the use of static networks across urban areas has significant cost, scaling, coverage, and performance issues

that will limit their deployment.

Leveraging people's mobile phones as mobile sensors to track mobile events in urban areas is an alternative approach that may overcome the problems of static sensor networks. However, there are several technical barriers in building mobile event tracking system using mobile sensors. First, mobile sensors need to be tasked before sensing. The mobile sensors that we refer to are the sensors that are attached to mobile phones. The primary purpose of these mobile phones is not sensing or tracking. The benchmark test presented in Chapter 4 shows that sensing is a task with a high energy requirement that will rapidly drain a cell phone battery. It is not desirable to drain the battery of mobile devices with a task that is not the primary purpose of the device. Hence, the mobile devices should be tasked only when and where the target is active and can be sensed.

Another issue is that the mobility of mobile phones (therefore, the mobile sensors) is uncontrolled. Because of the uncontrolled mobility of mobile sensors, there is no guarantee that there will always be a high enough density of mobile sensors around the target event. When there is limited density around the target event of interest, the mobile tracking system may lose the target. Recovering the target when the system loses track of the target due to the uncontrolled mobility is an important issue that must be addressed to enable event tracking using mobile phones.

1.4 Thesis Outline

In order to overcome the technical barriers presented above, we propose to use a combination of analysis, simulations, and experiments to better understand the problems and solution space. The outline of our study is as follows.

1.4.1 SWAN

Chapter 2 presents SWAN, a stateless network model that uses distributed control algorithms to provide service differentiation in mobile wireless ad hoc networks in a simple, scalable, and robust manner. The proposed architecture is designed to handle both real-time UDP (User Datagram Protocol) traffic and best-effort UDP and TCP traffic without the need for the introduction and management of per-flow state information in the network. SWAN supports per-hop and end-to-end control algorithms that primarily rely on the efficient operation of TCP/IP protocols. SWAN does not require the support of a QoS-capable MAC to deliver service differentiation. Rather, real-time services are built using existing best-effort, wireless MAC technology.

SWAN adapts the well-known additive increase and multiplicative decrease (AIMD) rate-control mechanism. In order to ensure that the bandwidth and delay requirements of real-time UDP traffic are met, rate control of TCP and UDP best-effort traffic is performed locally on every mobile node in a fully-distributed and decentralized manner. Rate control is designed to restrict best-effort traffic, yielding the necessary bandwidth required to support real-time traffic. Rate control also allows the best-effort traffic to efficiently utilize the bandwidth that is not utilized by the real-time traffic at any particular moment. The total rate of all best-effort and real-time traffic transported over each local shared media channel is maintained below a certain "threshold rate," limiting any excessive delays that might be experienced. SWAN rate control uses per-hop MAC delay as a feedback for local control.

SWAN uses sender-based admission control for real-time UDP traffic. The sender of a real-time session probes the network by sending a probe packet to the destination. The probe packet travels back to the sender with the traffic condition information that is required in making the admission decision of the real-time session. SWAN's sender-based admission control does not require that the intermediate node maintain the per-flow state. SWAN does not introduce excessive overhead because it does not send periodic signaling to maintain the

admitted real-time sessions. Instead, SWAN uses Explicit Congestion Notification (ECN) to dynamically regulate admitted real-time sessions in the face of network dynamics brought on by mobility or traffic-overload conditions. When the aggregated bandwidth of admitted real-time traffic exceeds the "threshold rate," the ECN mechanism forces the senders to re-establish or drop the real-time session of the sender.

The results from simulations, analysis, and experiments presented in Chapter 2 show that SWAN is capable of supporting real-time services with low and stable delays under various topology, traffic, and mobility conditions.

1.4.2 Funneling-MAC

In Chapter 3, we propose a localized, sink-oriented funneling-MAC that is capable of mitigating the funneling effect and boosting the fidelity of sensor network applications. The funneling-MAC is based on a CSMA/CA being implemented network-wide, with a localized TDMA algorithm overlaid in the funneling region (i.e., within a few hops from the sink) where the traffic is the most intense. In this sense, the funneling-MAC represents a hybrid MAC approach that does not have the scalability problems associated with the network-wide deployment of TDMA.

The funneling-MAC is "sink-oriented" because the burden of managing TDMA scheduling of sensor data transmission in the funneling region falls on the sink node, rather than on the resource-limited sensor nodes. We assume that the sink is likely to have more computational capability and energy reserves than simple sensors. The funneling-MAC is "localized" because TDMA only operates locally in the funneling region close to the sink and not across the complete sensor field. By using TDMA in this localized manner and putting more management burden on the sink instead of the sensors, the funneling-MAC offers a scalable solution for the deployment of TDMA in sensor networks.

The localized TDMA of the funneling-MAC is triggered by a beacon that is broadcasted

by the sink. The funneling-MAC integrates a light-weight clock synchronization embedded in the beacon messaging. The sink regulates the boundary of the intensity area (i.e., the area where the TDMA is performed) by controlling the transmission power of the beacon. The targeted size of the intensity area is calculated by using a dynamic, depth-tuning algorithm that enables the funneling-MAC to maximize the throughput and minimize packet loss at the sink point. The sink calculates the TDMA schedule for all nodes in the intensity area, and then the sink broadcasts the schedule to all nodes.

The sink broadcasts beacons periodically with high power. In order avoid interfering with such high-powered sink transmissions, the sensor nodes learn the superframe time details from the beacon messages. The nodes outside of the intensity region may not be aware of the funneling-MAC frame timing because they do not receive beacons. To avoid interference from such outside nodes, meta-schedule advertisement is embedded in the first event data packet that is transmitted in the TDMA schedule.

Through experimental results from two testbeds in different locations with different topologies (one with 45 mica-2 nodes and the other with 31 mica-2 nodes), we show that the funneling-MAC mitigates the funneling effect, improves throughput, decreases losses, and enhances energy efficiency. Importantly, the results show that the funneling-MAC significantly outperforms other representative MAC protocols, such as B-MAC, and more recent hybrid TDMA/CSMA MAC protocols, such as Z-MAC.

1.4.3 MetroTrack

In Chapter 4, we propose to use mobile sensors carried by people to track mobile events. We argue that sensor-enabled mobile phones are better suited for delivering sensing services, such as tracking in urban areas, than are the more traditional solutions, such as the static sensor network, which are limited due to scale, performance, and excessive cost. Increasingly, mobile phones are becoming more computationally capable and are equipped with a

variety of sensors, making a sensor network built on mobile phones more of a reality.

There are a number of challenges in building a mobile event tracking system using mobile sensors. First, mobile sensors must be tasked before sensing. Second, the design of the system is complicated by the fact that the mobility of people who carry the mobile sensors is uncontrolled. Finally, there is no guarantee that there will be sufficient density of mobile sensors around the target. This results in time-varying sensor coverage and disruptive tracking.

To address these challenges, we propose MetroTrack, a fully-distributed tracking system based on off-the-shelf mobile phones capable of tracking mobile targets through collaboration among local sensing devices that track and predict the future location of a target using a distributed Kalman-Consensus filtering algorithm. MetroTrack is predicated on the fact that a target will be lost during the tracking process, and thus it takes compensatory action to recover the target, allowing the tracking process to continue.

MetroTrack is based on two mechanisms, i.e., information-driven tasking and prediction-based recovery. Each sensor node independently determines whether to forward the tracking task to its neighbors or not, according to its local sensor reading state. Therefore, the tasking mechanism is information-driven and fully-distributed. If the sensor readings meet the criteria of the event being tracked, then the sensor node forwards the task to its neighbors to inform them that it detected the event. The recovery procedure is based on a prediction algorithm that estimates the lost target and its margin of error. MetroTrack adopts a geocast broadcasting scheme to forward the task to the sensors in the projected area of the target. The predicted geographical position of an event is computed using Distributed Kalman-Consensus Filtering (DKF) estimation. MetroTrack tracks a target based on local state and local radio interactions between mobile phones in the vicinity of the target and interacts with the back-end servers using cellular or infrastructure-based WiFi connectivity.

Chapter 4 presents the mathematical formulation of the prediction algorithm that provides the basis for recovery mechanism. In addition, a proof-of-concept prototype MetroTrack

System implemented using Nokia N80 and N95 phones is presented to show that it can effectively track a mobile noise source in an outdoor urban environment. Chapter 4 also presents the simulation study of the large-scale design space of MetroTrack, which cannot be analyzed from a small-scale testbed deployment. Simulation results indicate that MetroTrack is robust in the presence of different mobility models and device densities.

1.5 Thesis Contribution

The contribution of this dissertation is summarized as follows:

1. Chapter 2 presents a simple, distributed, and stateless network model called SWAN that supports service differentiation for real-time and best-effort traffic in wireless ad hoc networks. SWAN uses local rate control for UDP and TCP best-effort traffic and sender-based admission control for UDP real-time traffic. Explicit Congestion Notification (ECN) is used to dynamically regulate admitted real-time sessions in the face of network dynamics brought on by node mobility and traffic-overload conditions. Changes in topology and network conditions, even node and link failures, do not affect the operation of SWAN. An important contribution of Chapter 2 is that intermediate nodes do not keep any per-flow or aggregate state information in SWAN wireless networks. As a result, there is no need for the introduction of complex signaling or state control mechanisms to establish, update, refresh, and remove per-flow state information, as is the case with "stateful" QoS approaches. To the best of our knowledge, there has been no prior work on service differentiation in wireless ad hoc networks using stateless approaches. Another contribution is that SWAN does not require any QoS support from the MAC layer. SWAN assumes a best-effort MAC and performs feedback-based, control mechanisms to satisfy the bandwidth and delay requirements for real-time services.

2. In Chapter 3, we experimentally quantify the impact of the funneling effect in a sensor network. The results shown in Chapter 3 evidently represent the funneling effect and its debilitating impact on network performance. The results indicate that additional controls, such as scheduling in the network over the first few hops from the sink, could offer significant gains. Motivated by this result, we propose a localized, sink-oriented, funneling-MAC that explicitly recognizes the existence of the funneling effect in its design. While there have been several important new MAC protocols proposed for sensor networks, to the best of our knowledge, none has addressed the funneling effect. We show that the implementation of a simple hybrid TDMA/CSMA scheme in the intensity region, under the control of the sink, can significantly improve the throughput and loss performance of sensor networks. We also show experimentally that multiple MACs can coexist in the sensor networks. Specifically, a hybrid TDMA/CSMA that is operated in the funneling region (i.e., the area within a small number of hops from the sink) can seamlessly coexist with the pure CSMA outside of that region. In addition, any potential interference caused by dynamically increasing or decreasing the intensity region can be effectively managed by the funneling-MAC. Because TDMA only operates locally in the funneling region close to the sink and not across the entire sensor field, the funneling-MAC represents a hybrid-MAC approach that does not have a scalability problem.

3. Chapter 4 presents MetroTrack, a fully-distributed system capable of tracking mobile events using off-the-shelf mobile phones. To the best of our knowledge, this is the first sensor-based tracking system of mobile events using mobile phones. We identified a number of challenges in building a mobile-event tracking system using mobile phones. First, mobile sensors must be tasked before sensing can begin, and, for the system to scale effectively, only those mobile sensors near the target event should be tasked. Another challenge is that the mobility of mobile phones is uncontrolled,

and there is no guarantee that there will be sufficient density of mobile sensors around any given event of interest. As a result, targets will be lost and a means of efficiently recovering them is required. MetroTrack addresses these challenges with two mechanisms, namely, information-driven tasking and prediction-based recovery. The tasking procedure is information-driven because each sensor node independently determines whether to forward the tracking task to its neighbors or not, according to its local sensor reading state. The solution is therefore fully-distributed and uses local state only. The recovery procedure is based on a prediction algorithm that estimates the lost target and its margin of error. From the algorithmic perspective, we propose a novel distributed protocol for mobile sensors based on a distributed Kalman-Consensus filtering algorithm for recovering the lost target. We analyze the algorithm's performance by means of large-scale simulations using different mobility models and through the deployment of a real-world testbed using Nokia N80 and N95 phones.

Chapter 2

SWAN

2.1 Introduction

There is a growing need to support better than best effort quality of service (QOS) in mobile ad hoc networks [61], however, this is very challenging. Wireless ad hoc networks represent complex distributed systems, which interconnect wireless mobile nodes without the need for any fixed infrastructure. The interconnection between remote nodes relies on peer wireless and mobile nodes that operate as routers on behalf of source-destination pairs. Rerouting among mobile nodes causes network topology and traffic load conditions to change dynamically, making it difficult to support real-time applications with the appropriate QOS.

Another challenge in supporting QOS for real-time applications is associated with the design of the medium access control (MAC) protocol. The dynamic nature of wireless ad hoc networks makes it difficult to dynamically assign a central controller to maintain connection state and reservations. Because of this, best effort distributed MAC controllers are widely used in existing wireless ad hoc networks. The IEEE 802.11 Distributed Coordination Function (DCF) [67] is a good example of a best effort distributed MAC. There have been a number of proposals to support service differentiation at the MAC layer using distributed

control schemes [95], [93].

In this chapter, we take a practical approach that assumes a best effort MAC and propose a simple, distributed, and stateless network model called SWAN that uses feedbackbased control mechanisms to support real-time services and service differentiation in mobile ad hoc networks. SWAN uses local rate control for UDP and TCP best-effort traffic, and sender-based admission control for UDP real-time traffic. Explicit congestion notification (ECN) is used to dynamically regulate admitted real-time sessions in the face of network dynamics brought on by node mobility and traffic overload conditions. An important contribution of our work is that intermediate nodes do not keep any perflow or aggregate state information in SWAN wireless networks. As a result, there is no need for the introduction of complex signaling nor state control mechanisms needed to establish, update, refresh, and remove per-flow state information, as is the case with stateful QOS approaches found in the literature [61], [93]. Changes in topology and network conditions, even node and link failures, do not affect the operation of the SWAN control system. This makes the system simple, robust, and scalable. Instead of depending on state information, SWAN uses feedback information from the network. A rate control mechanism uses the per-hop MAC delay measurements from packet transmissions as feedback, while a source-based admission control mechanism uses rate measurements from aggregated real-time traffic as feedback.

In order to ensure that the bandwidth and delay requirements of real-time UDP traffic are met, rate control of TCP and UDP best effort traffic is performed locally at every mobile node in a fully distributed and decentralized manner. Rate control is designed to restrict best effort traffic yielding the necessary bandwidth required to support realtime traffic. Rate control also allows the best effort traffic to efficiently utilize the bandwidth that is not currently utilized by the real-time traffic at any particular moment. The total rate of all best effort and real-time traffic transported over each local shared media channel is maintained below a certain threshold rate, limiting any excessive delays that might be experienced. SWAN adopts engineering techniques that attempt to set the admission threshold rate at mobile

nodes to operate under the saturation level of the wireless channel based on insights from our earlier work on service differentiation support for wireless LANs [95]. To our knowledge, there has been little or no prior work on provisioning service differentiation in mobile ad hoc networks using stateless approaches.

SWAN is presented IEEE INFOCOM 2002 conference. Also, SWAN is published in IEEE Transcations on Mobile Computing (TMC) in 2002. SWAN is highly cited and has a significant impact on the research of mobile ad hoc networks and wireless sensor networks.

This chapter is structured as follows: The impact of SWAN is described in Section 2.2. The related work and motivation that underpins stateless wireless networks are discussed in Section 2.3. Section 2.4 describes a set of distributed control algorithms for rate control, source-based admission control, and dynamic regulation of real-time sessions that collectively constitute the SWAN network model. Section 2.5 analyzes the MAC delay and the busy probability of a wireless network with and without the SWAN rate control system. We show that, by controlling the probability of mobile nodes being in a backlogged state, the target MAC delay of the real-time traffic can be maintained. The results presented in this section confirm the feasibility and effectiveness of the SWAN approach. Sections 2.6 and 2.7 present a performance evaluation of SWAN using the ns-2 simulator and an experimental wireless testbed, respectively. The wireless testbed and ns-2 simulator source code are available from the Web [98]. Section 2.8 presents some concluding remarks.

2.2 Impact of SWAN

SWAN has a significant impact on the research of mobile ad hoc networks and wireless sensor network. In the survey [60] of MAC protocols for ad hoc wireless networks, SWAN is introduced as one of the prominant QoS service model for wireless ad hoc networks. SWAN is presented in IEEE INFOCOM 2002 conference and published in IEEE Transcations on Mobile Computing (TMC) in 2002. SWAN is cited by 411 research papers. In this section,

we present some of the papers that have cited SWAN.

SPEED [45] is one of the most influential work that cited SWAN. SPEED is a real-time communication protocol that provides three types of services, which are real-time unicast, real-time area-multicast and real-time area-anycast. SPEED follows the stateless approach just like SWAN. While SWAN provides traffic adaptation between the MAC and the network layer, SPEED provide both MAC adaptation and network adaptation. CACP [105] is an contention-aware admission control algorithm to support QoS in ad hoc networks. CACP follows SWAN's bandwidth admission control approach. CACP addresses that a node should consider both local resources and the resources of the neighbors in its carrier sensing range. In [105], the performance evaluation using ns2 network simulator has compared DSR, SWAN, and CACP. The results confirm that SWAN can provide decent real-time communication services over DSR. Shah et al. propose a dynamic bandwidth management scheme [90] that provides max-min fairness on a MAC protocol that does not provide weighted fair scheduling such as IEEE 802.11 DCF MAC. This work is influenced by SWAN in a sense that it support a feature that is not supported by the best effort MAC by providing a mechanism on top of the MAC. QPART [104] is a QoS protocol for real-time multimedia services in ad hoc networks. Like SWAN, QPART provides admission control and conflict resolution (similar to the regulation of SWAN) that relies only on the local information. QPART also compared its performance with SWAN using ns2 simulator. Xu et al. proposed a scalable QoS provisioning architecture [53] in ad hoc networks. This proposal follows SWAN's source-based admission control, local rate control, and ECN-based regulation. This proposal assume LANMAR [39] as the underlying routing protocol and utilizes the bandwidth information provided by LANMAR while SWAN do not assume any underlying routing protocol and gets the bandwidth information by probing.

As we have presented in this section, many research proposals succeeds the idea of SWAN and some of the proposals provide useful extensions to improve SWAN. Considering the resources of the neighbors in its carrier sensing range is one of the good ideas that we

may adopt. Since the SWAN have published, new technologies have become available. IEEE 802.11e EDCF MAC has emerged to provide prioritization among different class of services. SWAN can utilize EDCF to improve the performance of real-time services.

2.3 Related Work

A number of stateful QOS approaches have been discussed in the literature. In [64] and [85], multihop, multicluster packet radio network architectures support dynamic virtual circuit communications in an effective manner. In [61], an in-band signaling scheme manages per-flow soft-state in support of flow reservation (via hop-by-hop admission control) and restoration, and end-to-end flow adaptation. In [92], a coreextraction distributed ad hoc routing (CEDAR) algorithm is proposed that uses core extraction, link state propagation, and route computation to support QOS in wireless ad hoc networks. A ticket-based algorithm for QOS routing is discussed in [21]. State information maintained in the network, as proposed by all of these schemes, albeit hard or soft-state, is complex and problematic to manage in the face of mobility, and limits the scalability of these networks as the number of mobile nodes grows.

A number of papers found in the literature have proposed techniques that build on a combination of wellestablished algorithms to provide efficient traffic control in IP networks and, in some cases, wireless access networks. For example, additive increase multiplicative decrease (AIMD) [24], ECN [36], and fair queuing [29] have proven to be efficient components for such systems. While many of the proposals can provide some level of QOS support they are based on a set of architectural assumptions where all nodes in the network support state information, or implement a certain set of end-to-end control algorithms, or require the support of QOS-capable MACs at each node along the path, or combine architectural components in such a way that they operate efficiently only if glued together according to the proposed architecture. In this chapter, we take a more pragmatic approach and argue

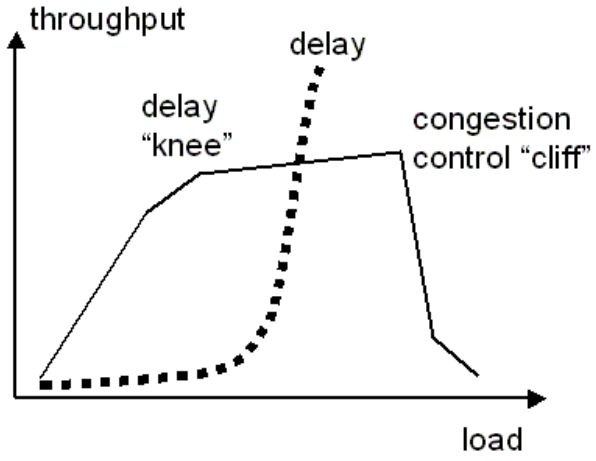


Figure 2.1: General behavior of a congestion controlled system.

that it is only necessary to consider the MAC QOS [93] support offered locally at individual nodes, if available. There is no need for end-to-end QOS mechanisms typified by the stateful and monolithic architectures found in the literature. Rather, SWAN simply assumes a best effort MAC, and proposes a simple, distributed, and stateless network model that uses feedback-based control mechanisms and local control to support real-time services and service differentiation. We argue that control mechanisms for best-effort traffic should be distributed and of local scope. It is not feasible to efficiently distribute prompt information to the edges of the system in order to protect real-time traffic, particularly not in ever-changing mobile ad hoc networks. Nevertheless, the local control has to rely on the existence of independent, end-to-end algorithms that can sense and react to the distributed, local actions. The most important of such algorithms are TCP with or without ECN, and end-toend congestion control for UDP-based applications (e.g., based on AIMD or equation-based rate control).

SWAN adapts the well-known AIMD rate control mechanism to address some of these challenges. AIMD algorithms are widely used by a number of transport protocols. For example, the TCP congestion control mechanism uses AIMD window-based control, while WTCP [92] uses AIMD rate control. In [7], AIMD control is applied to real-time UDP traffic. TCP and WTCP use AIMD control to improve the performance of TCP traffic. In

contrast, SWAN uses AIMD rate control to improve the performance of real-time UDP traffic. TCP attempts to avoid network congestion collapse by using packet loss as feedback. We propose to control the rate of TCP traffic more conservatively to avoid excessive delays of real-time UDP traffic by using local per-hop packet delays as a feedback to local rate controllers. Figure 2.1 illustrates the general behavior of a TCPlike congestion controlled system [24]. The congestion control algorithm ensures that the system operates around, or preferably close to the cliff, which ensures maximum system throughput, but at the cost of larger queues, and therefore larger average packet delays. The SWAN AIMD control algorithm discussed in this chapter, on the other hand, keeps the system at the delay knee where the system throughput is almost the same as at the cliff, but the buffers are significantly less loaded, so the delay is close to the minimum. SWAN achieves this by using the per-hop MAC delay as a feedback for local control instead of packet loss. The reason for doing this is that loss typically happens at the cliff, while delays start to increase at the knee, as illustrated in Figure 2.1.

In [12], [18], [9], [58], endpoint admission control schemes that use measurement-based admission control are proposed and analyzed for wireline networks. End hosts probe the network by sending probe packets at the required data rate, monitoring the level of packet losses [18], [10], or ECN congestion marks [58]. These endpoint admission control approaches do not require that intermediate routers maintain per-flow state. While these approaches clearly address the needs of scalability and service quality in wireline networks, they do not address the important issue of mobility, which complicates the delivery of QOS in mobile networks. If such schemes were to be implemented in mobile networks, then the admission control guarantees would be continuously violated by mobility as admitted flows are rerouted in wireless networks. SWAN resolves this issue by performing ECN-based regulation for admitted flows. Another problem relates to the overhead generated by the periodic probing associated with these schemes, which is driven at the required data rate of new flows. Such an approach is unsuitable for limited bandwidth wireless networks. SWAN

does not send any periodic signaling, resolving this problem.

Traffic is highly unpredictable and bursty in nature, particularly if we consider networks with relatively small levels of aggregation such as wireless ad hoc networks. We believe that only a pragmatic solution to service differentiation is feasible and likely to be successful. SWAN assumes that most of the network capacity will be utilized by besteffort traffic, which can serve as a buffer zone or absorber for real-time traffic bursts introduced by mobility (e.g., because of the rerouting of admitted real-time sessions) or traffic variations (e.g., bursty data). Allowing best effort traffic to act as a buffer zone for real-time traffic, in particular, allowing it to absorb unpredictable bursts of real-time traffic found in mobile networks, allows SWAN to shift the emphasis away from precise hop-by-hop admission control, which, historically, is very difficult to do well and in the end not necessary. In SWAN, we assume that there will be always best-effort traffic present that can be locally and rapidly rate controlled in an independent manner at each node to yield the necessary low delays and stable throughput for real-time traffic. In contrast to admission control, which can be coarse in SWAN, local rate control of best effort traffic at each mobile node has to be very efficient, prompt, robust, and precise. While we argue rate control of best effort traffic should be local and fast, the admission control of real-time traffic should work end-toend since most of the real-time applications do not implement congestion control. For such applications, there must be a regulation algorithm that works at the edge nodes preventing the emission of nonresponsive flows.

2.4 Distributed Control Algorithms

The SWAN model includes a number of mechanisms used to support rate regulation of best effort traffic, as illustrated in Figure 2.2. A classifier and a shaper operate between the IP and best effort MAC layers. The classifier is capable of differentiating real-time and best effort packets, forcing the shaper to process best effort packets but not real-time packets. The

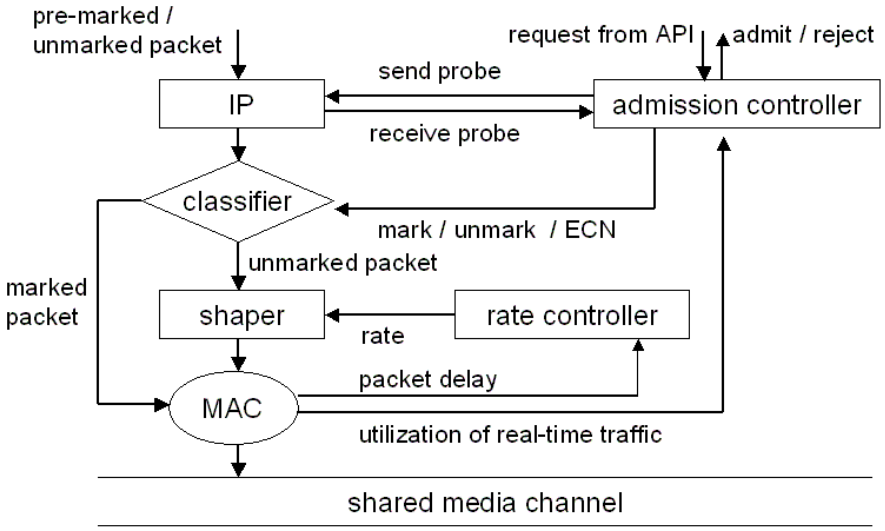


Figure 2.2: SWAN model.

shaper represents a simple leaky bucket traffic shaper. The goal of the shaper is to delay best effort packets in conformance with the rate calculated by the rate controller. When a session is admitted there is no admission control decision taken at intermediate nodes. Rather, the admission control test to determine if a new real-time session should be admitted or not is conducted solely at the source node based on the result of an end-to-end request/response probe. A key operation of the admission controller is to efficiently estimate local bandwidth availability. Typically, a probe is sent at the beginning of a session or, as discussed later, when mobility or channel load conditions force an admitted real-time session to reestablish its end-to-end service quality. In what follows, we describe the SWAN distributed control algorithms.

2.4.1 Local Rate Control of Best Effort Traffic

Each node in the mobile ad hoc network independently regulates best effort traffic. The rate controller determines the departure rate of the shaper using an AIMD rate control algorithm based on feedback from the MAC. This feedback measure used by the rate controller

```
Procedure update_shaping_rate ( )
/* called every T second period */
Begin

if (n > 0)                      /* one or more packets have delays
                                   greater than the threshold delay d sec */
   s ← s * (1 – r / 100)        /* multiplicative decrease by r% */

else

   s ← s + c                    /* additive increase by c Kbps */

if ( (s – a) > a * g / 100)     /* difference between actual rate and shaping
                                   rate is greater than g% of actual rate */
   s ← a * (1+ g / 100)         /* adjust shaping rate to match actual rate */

end
```

Figure 2.3: SWAN AIMD rate control algorithm.

represents the packet delay measured by the MAC layer. As mentioned previously, SWAN assumes best effort MAC technology. Packet delay for the IEEE 802.11 DCF mode MAC, for example, can be measured rather simply. When a packet arrives at the MAC layer, the MAC listens to the channel and defers access to the channel according to the carrier sense multiple access with collision avoidance (CSMA/CA) algorithm. When the MAC acquires access to the channel, then RTS-CTS-DATA-ACK packets are exchanged. The reception of an ACK packet at the transmitter indicates that a packet is received successfully. The packet delay represents the time it took to send the packet between the transmitter and next-hop receiver including the total deferred time (including possible collision resolution) plus the time to fully acknowledge the packet. This is simply measured at the source node by subtracting the time that a packet is passed to the MAC layer (from the upper layer) from the time an ACK packet is received from the next-hop receiver.

The SWAN AIMD rate control algorithm is shown in Figure 2.3. Every T seconds, each mobile host increases its transmission rate gradually (additive increase with increment rate of c Kbps) until the packet delays become excessive. The rate controller detects excessive

delays when one or more packets have greater delays than the threshold delay d sec. As soon as the rate controller detects excessive delays, it backs off the rate (multiplicative decrease by r percent). The threshold delay d is based on the real-time delay requirements of applications in wireless network, as discussed by our previous work [95]. The shaping rate is adjusted every T seconds. The period T should be small enough to be responsive to the dynamics of mobile ad hoc networks [78]. If there is a large difference between the shaping rate and the actual transmission rate, then a mobile host is capable of transmitting a burst without due control, potentially limiting the performance of real-time traffic. To resolve this problem, the rate controller monitors the actual transmission rate. When the difference between the shaping rate and the actual rate is greater than g percent of the actual rate, then the rate controller adjusts the shaping rate to be g percent above the actual rate. This gap (i.e., g percent) allows the best-effort traffic to increase its actual rate gradually.

In this chapter, we argue that bandwidth and delay bound requirements of real-time traffic can be adequately supported by using rate control based on our simple SWAN AIMD rate control algorithm, while best effort traffic can efficiently utilize any remaining bandwidth. However, to fully support real-time traffic, local rate control of best effort traffic is insufficient. There is also a need to support admission control.

2.4.2 Source-Based Admission of Real-Time Traffic

Using a shared wireless channel allows mobile hosts to listen to packets sent within their radio transmission range. An admission controller uses this feature to measure local resource availability. At each node, the admission controller measures the rate of real-time traffic in terms of bits per second. Note that in order to smooth out small-scale traffic variations, the admission controller uses a running average (e.g., weighted moving average) of these measures. If we know the threshold rate [95] that would trigger excessive delays, then bandwidth availability in a local shared media channel is simply the difference between the

threshold rate and the current rate of the real-time traffic. However, it is difficult to estimate the threshold rate accurately because the threshold rate may change dynamically depending on traffic patterns. It is not desirable to admit real-time traffic up to the threshold rate for a number of reasons. First, best effort traffic would be starved of resources should the real-time traffic consume bandwidth up to such a threshold rate. Best effort traffic is rate controlled to yield the bandwidth required for real-time traffic and to keep the total traffic, both real-time and best effort, below the threshold rate. Second, there would be no flexibility to tolerate channel dynamics, as previously discussed. The total rate of aggregated real-time traffic may be dynamic due to changes in traffic patterns and host mobility. Due to host mobility, for example, intermediate nodes may need to maintain real-time traffic in excess of resources set-a-side for real-time traffic. There are a number of possible ways to address this issue. We take a simple approach and admit real-time traffic up to a rate that is more conservative than the threshold rate. We consider the estimated available bandwidth of a local shared media channel to be the difference between this conservative admission control rate and the current rate of the realtime traffic. With such a policy, we can use fixed, coarse, and statistically approximated values for the admission control rate. Even though the measure is conservative, the utilization of the network is not limited because any remaining unutilized bandwidth will be potentially absorbed by the best-effort traffic. This approach is simple and flexible and allows bandwidth sharing between real-time and best-effort traffic in an efficient manner. The process for admitting a new real-time session is as follows: The admission controller located at the source node sends a probing request packet toward the destination node to estimate the end-to-end bandwidth availability. The probing request packet is a UDP control packet that contains a bottleneck bandwidth field. Each intermediate node between the source-destination pair intercepts the probing request packet and updates the bottleneck bandwidth field in the packet if the bandwidth availability at the node is less than the current value of the field. Therefore, if the local bandwidth availability is different for each hop along the path between the source and destination hosts, then the value of the bottleneck

field at the destination node represents the bottleneck bandwidth found along the path. The destination node sends a probing response packet back to the source node with the bottleneck field copied from the probing request message received by the destination node. There is no need for this probe response message to follow a reverse path back toward the source node. Once the source node receives the probing response packet, it can execute the simple source-based admission control test by comparing the end-to-end bandwidth availability and the bandwidth requirement for the new real-time session. Note that no bandwidth request is carried in the probe message, no admission control is executed at any intermediate node, nor are there any resources allocated or reserved on behalf of the source node during the lifetime of an admitted session. Rather, the probe instantaneously reads the state of the network path presented to it by the routing protocol and makes a local source-based admission decision based on the probe response. What makes such a stateless approach work is that all nodes independently regulate best effort traffic and each source node uses admission control for real-time sessions. When a new real-time session is admitted, the packets associated with the admitted flow are marked as RT (real-time packets/traffic). The classifier looks at the marking and, if the packet is marked as RT, the packet will bypass the shaper mechanism, remaining unregulated. Here, there is an implicit assumption that a source node regulates its real-time sessions based on its admission control decision.

2.4.3 Dynamic Regulation of Real-Time Traffic

Impact of Mobility and False Admission

Mobility and false admission represent two conditions that violate this simple approach to source-based admission control, complicating the delivery of service assurances. Host mobility is harmful to service assurances because realtime flows admitted along a certain path can be dynamically rerouted. Because nodes are unaware of flow rerouting due to mobility, resource conflicts can arise and persist. Source nodes, for example, that have

previously admitted flows are unaware of host mobility and the rerouting of flows through new intermediate nodes that may have insufficient resources to support previously admitted realtime traffic. False admission is the result of multiple source nodes simultaneously initiating admission control at the same instance and sharing common nodes between sourcedestination pairs. Because intermediate nodes do not maintain state information and admission control is conducted at the source node in a fully decentralized manner, each source node may receive a response to their probe message indicating that resources are available when, in fact, they are not. However, the source node being unaware of this fact falsely admits a new flow and starts transmitting real-time packets under the assumption that resources are available to meet the flows needs. Consider the following simple false admission scenario. Four source nodes want to establish video flows at a rate of 200 Kbps for each flow and start probing the network. A common intermediate node only has resources to support 200 Kbps of real-time traffic, sufficient to support only a single video flow. However, in the case of false admission, all the flows are admitted erroneously because all the nodes see a reservation that can support 200 Kbps each. This results in the four source nodes injecting data into the wireless network with an aggregate rate of 800 Kbps, destined toward the common node under discussion. If left unresolved, the rerouting of admitted real-time flows can cause excessive delays in realtime traffic because the utilization of the admitted or falsely admitted real-time traffic can violate the admission control rate potentially exceeding the threshold rate by a significant margin. To resolve this problem, we augment the SWAN AIMD rate control and source-based admission control algorithms with dynamic regulation of real-time traffic when congestion/overloading is experienced by nodes due to the rerouting of admitted flows or false admission.

Regulation Algorithms

The ECN-based regulation of real-time sessions operates as follows: Each node continuously, and independently, measures the utilization of its real-time traffic to estimate the local available bandwidth, as discussed earlier. Each mobile node can detect violations (i.e., congestion/overload conditions) using this periodic traffic measurement. When a node detects such a violation, it starts marking the ECN bits in the IP header of the real-time packets. The destination node monitors the ECN bits and notifies the source using a regulate message. When the source node receives a regulate message, it initiates reestablishment of its real-time session based on its original bandwidth needs. To reestablish a realtime session, a source node follows the same process as setting up a new session by sending a probing request toward the destination. A source node terminates the session if the estimated end-to-end bandwidth indicated in the probing response packet cannot meet its existing session needs. This is one of the reasons why we call our approach to service differentiation in mobile ad hoc networks soft because an admitted real-time flow may encounter both periodic violations in its bandwidth requirements and, in the worst case, may have to be dropped or live with degraded best effort delivery. If the nodes in a congested or overloaded condition were to mark all packets with CE (Congestion Experienced), then all sessions traversing these nodes would be forced to reestablish their real-time service at the same instance. Such an approach is inefficient and would lead to erroneous behavior of the communications system. For example, all sources may reprobe the network, see network resources over utilized and drop all their flows accordingly. This clearly is not the best policy. More systematic approaches may only penalize a small number of sources. To address the problem, we consider two approaches and analyze their suitability and trade-offs.

Source-Based Regulation: When an intermediate node experiences overloaded or congested conditions it marks all flows with CE. When destination nodes encounter packets with the CE bit marked they send regulate messages to the appropriate source nodes to force

the reestablishment of flows that have previously been successfully admitted. However, in this case, the source node does not immediately initiate reestablishment upon receipt of a regulate message. Rather, it waits for a random amount of time before initiating the reestablishment procedure. Under such a regime, source regulation would be staggered, thereby avoiding flash-crowd conditions where a number of sources simultaneously initiate regulation (i.e., reestablishment of service) at the same time, see the path overbooked and drop their real-time sessions accordingly. Under a staggered regime, the rate of the real-time traffic will gradually decrease until it reaches below the admission control rate. At that point, congested or overbooked nodes will stop marking packets. Because flows can be admitted by mistake, due to false admission, source nodes need to be capable of differentiating between regulation associated with false admission and regulation due to mobility. Nodes can do this by keeping some state information about newly admitted flows versus on-going flows. This allows a source node to take immediate corrective action in the case where it receives a regulation message for a flow that it just admitted, albeit falsely. A disadvantage of this approach is that sources that regulate earlier than other sources (i.e., wait the shortest period of time before initiating reprobing/ reestablishment) are more likely to find the path overbooked and be forced to drop their sessions. An advantage of this scheme, however, is that it is purely source-based.

Network-Based Regulation: Rather than marking all packets with CE, congested/overloaded nodes randomly select a congestion set of real-time sessions and only mark packets associated with the set. This can be done using a hash function without keeping any per-flow state at the intermediate nodes. A congested node marks the congested set for a period of time T seconds and then calculates a new congested set. As in the case of the previous algorithm, nodes stop marking packets congested when the measured rate of the real-time traffic drops below the admission control rate. Under such an approach the rate of the realtime traffic will gradually decrease until it reaches below the admission control rate. However, there is a need for intermediate routers to distinguish between flows that have been falsely admitted or

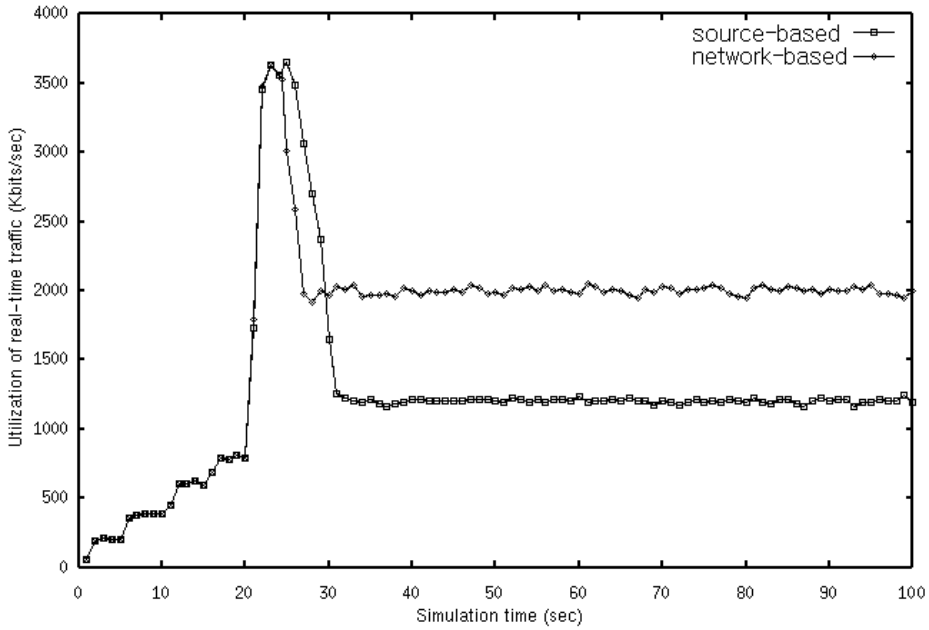


Figure 2.4: Comparison of the source-based delayed regulation and network-based set regulation schemes Trace of the actual rate of the real-time traffic for both schemes.

not. In this case, source nodes could use an additional bit in the TOS field to indicate if a RT session is new or old (namely, RT-new, RT-old). When a flow is newly admitted, packets are marked as RT-new for a period of time before being marked as RT-old. A disadvantage on this scheme is that it requires some intelligence at intermediate nodes to manage the congested sets and determine if a flow is new or old in order to correctly respond to false admission.

Performance Considerations and Trade-Offs

There are a number of trade-offs between the source-based and network-based regulation schemes. Figure 2.4 compares the two approaches and shows how a combination of admission control and regulation can manage the rate of real-time traffic under overload conditions. The results are obtained from an ns-2 simulation of SWAN that is further discussed in Section 2.6. To observe how regulation works, we consider an extreme scenario where a number of real-time sessions are suddenly rerouted in the network through a certain

hop. The rerouted sessions have all previously been successfully admitted. As shown in Figure 2.4, due to mobility, rerouted flows start to be routed over the shared media channel (without the reapplication of admission control) at 20 sec into the simulation scenario. Also, a number of source nodes simultaneously perform admission control for new real-time sessions creating conditions for false admission at 20 sec into the scenario. Note that the SWAN admission control mechanism allows small-scale violation up to the threshold rate for ECN-based regulation. In this simulation, the admission control rate of the real-time traffic is set at 2 Mbps and the threshold rate is 3.5 Mbps. The channel bandwidth is 11 Mbps. For a more detailed discussion on the setting of these system parameters, which are determined by considering the requirements of the admission control and threshold rates, see our previous work [95] on the analysis of distributed MAC delays. As soon as the rate exceeds the threshold rate, the mobile nodes in the shared media channel begin to mark the ECN field of the packets of all real-time flows (in the case of source-based regulation) or the active set (in the case of network-based regulation). Figure 2.4 shows the results for both schemes. For both algorithms, flows are dropped gradually after intermediate nodes start to mark the ECN field in the packets of real-time flows. Note that the dropped flows may not necessarily be the rerouted sessions but existing sessions that were admitted previously. This may seem unfair but our approach is stateless and there is no mechanism for congested nodes to differentiate between existing and rerouted sessions. Furthermore, it is likely that most realtime sessions would be rerouted multiple times during the lifetime of their sessions. In this case, there is little benefit in attempting to discriminate between the existing and rerouted flows when statistically all sessions should be treated in the same manner on average. We used a random number between (0, 7) to support the back-off and set selection functions required by the source-based and network-based regulation schemes, respectively. From Figure 2.4, we observe that the rate of the real-time traffic is gradually reduced until it reaches the admission control rate for the network-based regulation approach. We observe that this took approximately four seconds for network-based regulation. The

simulation results indicate that after rerouting, 19 sessions were traversing the node under congested conditions. Five of these sessions were traversing the node prior to the overload condition. Eight sessions were rerouted sessions and six sessions were the product of false admission. The regulation process resulted in 11 sessions being successfully regulated with eight sessions being dropped. In the scenario discussed above, all falsely admitted sessions were dropped prior to other flows being dropped, with one of the dropped flows traversing the node before flows were rerouted. The response of the source-based regulation scheme took two seconds longer than the network-based approach. This additional latency caused more real-time traffic to be dropped over that experienced by network-based regulation. In the case of source-based regulation, the network operates below the admission control rate indicating that the scheme may result in under utilization of resources at the congested node in the worst-case scenario, as shown in Figure 2.4. In summary, network-based regulation performs better than source-based regulation with the cost of an additional 1-bit in the packet header for marking flows new/old, and more intelligence at intermediate nodes to manage the sets to mark.

2.5 Analysis

We analyze the MAC delay and the probability that mobile hosts find themselves in a backlogged state in IEEE 802.11 ad hoc wireless networks. We use the terms SWAN and DCF to refer to DCF wireless networks with and without SWAN, respectively. DCF is widely used in wireless LAN and wireless ad hoc networks, and uses CSMA/CA. In the DCF mode, a mobile host must sense the medium before initiating the transmission of a packet. If the medium is sensed as being idle for a distributed interframe space (DIFS) period, then the mobile host can transmit a packet. Otherwise, transmission is deferred and a backoff process is entered. In the backoff process, the mobile host computes a random value in the range of 0 to the contention window (CW). A backoff time interval is computed as

this random value multiplied by the slot time. This backoff interval is then used to initialize the backoff timer. This timer is decreased only when the medium is idle. As soon as the backoff timer expires, the mobile host transmits a packet. In our previous work [95], we modified the DCF algorithm to support service differentiation by using different minimum contention windows (CW_{min}) for different priority classes. We use the term CW_{min} to refer to this modified DCF algorithm compared with DCF and SWAN discussed above. In this section, CW_{min} is chosen to be 15 slots for Class 1 and 31 slots for Class 2 mobiles. In Section 2.5.1, we show through analysis that SWAN performs better than DCF and CW_{min} in terms of MAC delay. Section 2.5.2 explains why this is the case. We show that, backlogged state, the targetMACdelay of the real-time traffic can be maintained. This result confirms that the SWAN approach is feasible and effective.

2.5.1 Analysis of MAC Delay

Assume there are two classes of mobile hosts in a shared channel environment. Class 1 and Class 2 represent real-time UDP traffic and best effort TCP traffic, respectively. Each of the n_1 Class 1 mobile hosts have an active UDP session, and each of the n_2 Class 2 mobile hosts have an active TCP session.Wedefine an idle mobile host as a mobile host whose MAC layer is idle and interface queue is empty. If a mobile host is not idle, then it is busy. Denote the portion of time that a class i mobile host is busy as $p_{on,i}$. From [9], a busy class i mobile hosts transmission probability in each time slot is represented as,

$$\tau_i = \frac{2 \cdot (1 - 2p_i)}{(1 - 2p_i)(W + 1) + p_i W(1 - (wp_i)^m)}, \tag{2.1}$$

where p_i is the collision probability for a class i mobile host at each time slot. $W - 1$ is the initial back-off window, and $W2^m - 1$ is the maximum back-off window in the IEEE 802.11 DCF protocol. By following the procedure in [9], the collision probability can be

represented as,

$$p_1 = 1 - (1 - p_{on,1}\tau_1)^{n_1-1}(1 - p_{on,2}\tau_2)^{n_2},$$

$$p_2 = 1 - (1 - p_{on,1}\tau_1)^{n_1}(1 - p_{on,2}\tau_2)^{n_2-1}. \tag{2.2}$$

The probability that one or more packets are sent to the channel at each slot is then,

$$P_{tr} = 1 - (1 - p_{on,1}\tau_1)^{n_1}(1 - p_{on,2}\tau_2)^{n_2}, \tag{2.3}$$

and the probability of a successful transmission each slot, $P_s = P_{s_1} + P_{s_2}$ is

$$P_s = P_{s_1} + P_{s_2} = \frac{n_1 p_{on,1}\tau_1(1 - p_{on,1}\tau_1)^{n_1-1}(1 - p_{on,2}\tau_2)^{n_2}}{P_{tr}},$$

$$+ \frac{n_2 p_{on,2}\tau_1(1 - p_{on,1}\tau_1)^{n_1}(1 - p_{on,2}\tau_2)^{n_2-1}}{P_{tr}}, \tag{2.4}$$

The total throughput of the system (in packets/sec) can be represented as,

$$S_i = \frac{P_{si}P_{tr}}{(1 - P_{tr})\sigma + P_{tr}(P_s T_s + (1 - P_s)T_c)},$$

$$S = S_1 + S_2, \tag{2.5}$$

where σ is the length of a time slot, which is 20 microseconds in all our simulations. T_s and T_c are the times needed to send un-collided and collided packets, respectively, on the channel. T_s and T_c are calculated from the packet length distribution taking into account the overhead of the MAC and physical layers (i.e., SIFS, DIFS, ACK, header, and preamble). The length of collided packets is approximated as the maximum length of two collided packets. The overall average MAC delay, and delays for each class, can be simply calculated using Littles formula as,

$$d = \frac{p_{on,1}n_1 + p_{on,2}n_2}{S},$$

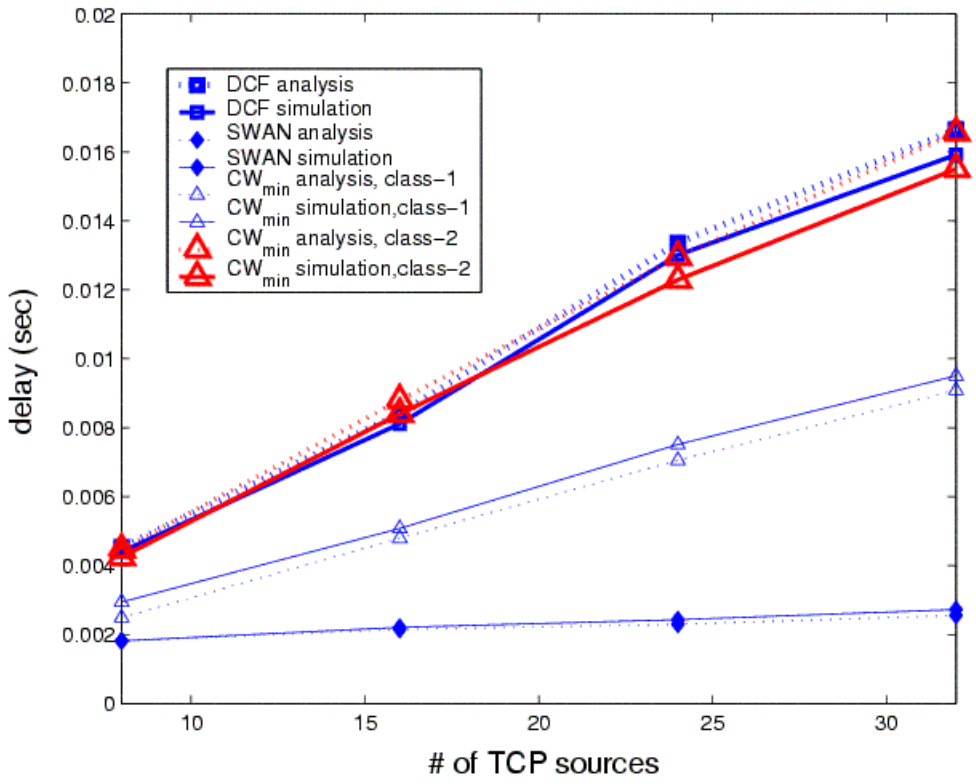


Figure 2.5: Comparison of packet delay.

$$d_i = \frac{p_{on,i} n_i}{S_i}. \tag{2.6}$$

We carried out an ns-2 simulation for four video (Class 1) mobile hosts and eight 32 FTP (Class 2) mobile hosts. In this simulation, video sessions are modeled as CBR sources and the FTP sessions have infinitely long file sizes that last for the whole simulation period. We denote the uncontrolled system as DCF, and the system with the proposed feedback control as SWAN. Because it is difficult to use a simple model to characterize the flow control of TCP/IP, coupled with a queuing system on top of the MAC layer, and the MAC layer and traffic shaper (for SWAN), we record the quantity $p_{on,1}$, $p_{on,2}$ during the simulation as an approximation of this complex system. With $p_{on,1}$, $p_{on,2}$, n_1, and n_2 known, we jointly solve Equations 2.1, 2.2 for p_1, p_2, τ_1, and τ_2, then from Equations 2.3, 2.4, 2.5, 2.6, the average delay is computed for each of the systems. The results are plotted in Figure 2.5. In the case of DCF and SWAN, there is no service differentiation at the MAC layer so the average MAC

delays shown in Figure 2.5 represent overall average delay for both classes. Analytical results shown are calculated from d_i in 2.6. In the CW_{min} case, the MAC layer supports service differentiation, so Class-1 and Class-2 average delays are show in Figure 2.5.

2.5.2 Analysis of Busy Probability

We now analyze SWAN from another perspective and try to find the value of $p_{on,2}$, (i.e., the probability of a Class 2 mobile host being in a backlogged state) that Class 2 mobile hosts must achieve so that the target average MAC delay can be maintained. We assume no packet loss due to buffer overflow for Class 1 mobile hosts. Because Class 1 mobile hosts carry UDP real-time sessions with known data rates (S_1 packets/sec) from the application layer, we set our target MAC delay as d, so $p_{on,1}$ can be acquired from the following:

$$p_{on,1} = d \cdot S_1. \tag{2.7}$$

We use a procedure, called Equilibrium Point Analysis (EPA) [72], in calculating the collision probabilities. In this approach, probabilities are calculated in terms of the equilibrium point of the system, and the average number of mobile hosts in each state is often chosen as the point. Here, we choose the average number of backlogged Class 2 mobile hosts $\bar{n}_2$ as the equilibrium point. We modify Equation 2.2 as,

$$p_1 = 1 - (1 - p_{on,1}\tau_1)^{n_1-1}(1 - \tau_2)^{\bar{n}_2},$$

$$p_2 = 1 - (1 - p_{on,1}\tau_1)^{n_1}(1 - \tau_2)^{\bar{n}_2-1}. \tag{2.8}$$

Note that we can still use Equation 2.2 to find $p_{on,2}$ but slightly worse analytical results are produced. We modify Equation 2.3 accordingly as,

$$P_{tr} = 1 - (1 - p_{on,1}\tau_1)^{n_1}(1 - \tau_2)^{\bar{n}_2}. \tag{2.9}$$

The probability of transmission of a Class 1 mobile host being successful, conditioned on at least one mobile host transmitting, is given by,

$$p_{s_1} = \frac{n_1 p_{on,1}\tau_1(1-p_{on,1}\tau_1)^{n_1-1}(1-\tau_2)^{\bar{n}_2}}{P_{tr}}. \tag{2.10}$$

The probability of a packet transmission being successful, conditioned on at least one mobile host transmitting, is then given by,

$$P_s = \frac{n_1 p_{on,1}\tau_1(1-p_{on,1}\tau_1)^{n_1-1}(1-\tau_2)^{\bar{n}_2}}{P_{tr}}$$

$$+ \frac{\bar{n}_2\tau_1(1-p_{on,1}\tau_1)^{n_1}(1-\tau_2)^{\bar{n}_2-1}}{P_{tr}}. \tag{2.11}$$

As in Equation 2.5, the throughput (in packets/sec) of Class 1 mobile host is,

$$S_1 = \frac{P_{s_1}P_{tr}}{(1-P_{tr})\sigma + P_{tr}(P_s T_s + (1-P_s)T_c)}. \tag{2.12}$$

We choose the Class 1 MAC delay in SWAN observed in the simulation-video curve shown in Figure 2.5 as our target delay d, and CBR rates as s_1, then $p_{on,1}$ is calculated from Equation 2.7. We put Equations 2.7, 2.9, 2.10, 2.11 to Equation 2.12 and solve Equations 2.1, 2.8, 2.12 jointly for $\bar{n}_2$, τ_1, τ_2, p_1, and p_2. Following this, we calculate the probability of a Class 2 mobile host being busy as,

$$p_{on,2} = \frac{\bar{n}_2}{n_2}. \tag{2.13}$$

Figure 2.6 shows analytical results for $p_{on,2}$ in comparison to the measured $p_{on,2}$ from our simulation of SWAN. In order to achieve the target delay d for Class 1 mobile hosts, we need to keep the probability of Class 2 being busy to be less than $p_{on,2}$. The input parameters of the above analysis are d (desired average delay), S_1 (the throughput of Class

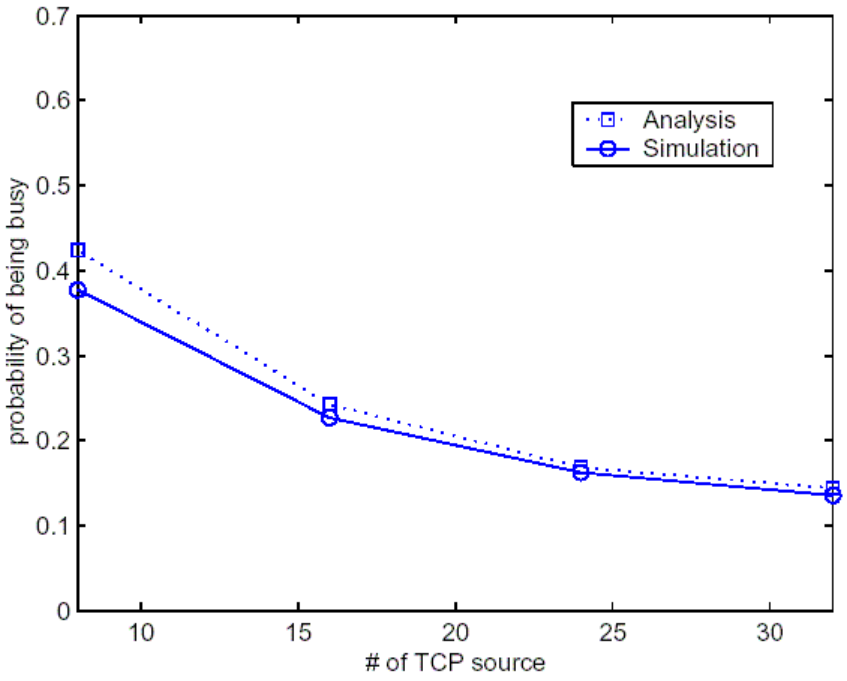


Figure 2.6: Comparison of busy probability.

1 mobile hosts), n_1, and n_2 (the number of Class 1 and Class 2 mobile hosts, respectively) and the output is $p_{on,2}$ (the probability of Class 2 mobile hosts being busy). With S_1, n_1, n_2 fixed, the delay statistics are positively related to $p_{on,2}$, where a delay violation implies a violation of $p_{on,2}$. This prompts us to correct this situation by minimizing the possibility that a class 2 mobile host will be still in a backlogged state. In SWAN, this is achieved by the multiplicative decrease procedure when the delay violation is observed. On the other hand, when the delay is small, $p_{on,2}$ is also small, and the system is under-loaded. The additive increase procedure increases $p_{on,2}$ by gradually increasing the packet arrival rate. The AIMD mechanism with delay feedback is thus an automatic procedure to keep $p_{on,2}$ on a desirable level, so the bandwidth is effectively utilized, but the system is not overloaded. As a result, TCP traffic has reasonable throughput, while UDP traffic achieves the desired delay performance. Figure 2.6 compares the analytical result of the desired $p_{on,2}$ for achieving the target MAC delay, with the measurement from the simulations of SWAN. The simulation curve closely matches the analytical curve. This result confirms that AIMD rate control is

capable of keeping $p_{on,2}$ at a desired level, and thus maintaining the target MAC delay.

2.6 Evaluation

We implemented SWAN using the ns-2 simulator and its wireless extensions developed at Carnegie Mellon University (CMU) [13]. The SWAN ns-2 extensions include the AIMD rate controller, admission controller, packet delay measurement mechanism, local utilization monitoring, probe protocol for bandwidth availability estimation, and ECN. The ns-2 SWAN simulation code is available on the Web [98]. In what follows, we evaluate and compare the performance of DCF, SWAN, and CW_{min} [95]. Throughout the simulation, each mobile host has a transmission range of 250 meters and shares an 11 Mbps radio channel with its neighboring nodes. The simulation includes a two-ray ground reflection model and IEEE 802.11 MAC protocol.

2.6.1 Performance of a Single Shared Channel

To best understand the characteristics of the SWAN rate control and admission control mechanisms, we first study a wireless ad hoc network that comprises a single shared wireless channel. The simulated network has a square shape of 150m x 150m where all wireless ad hoc mobile nodes share a single radio channel of 11 Mbps. The source and destination nodes associated with flows are distributed among the mobile nodes in the wireless ad hoc network. We ran a large set of simulations using different values for the AIMD rate controller parameters, c (increment rate, Kbps/sec), r (decrement rate, percent), and g (gap between actual rate and shaping rate, percent) to understand the characteristics, trade-offs, and performance of our rate control mechanism.

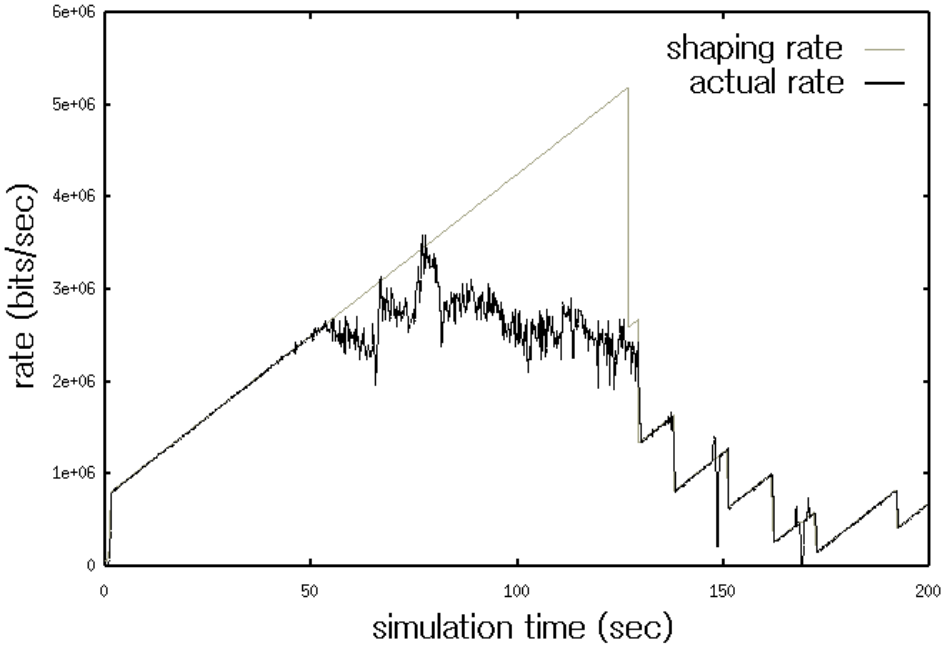


Figure 2.7: Trace of the shaping rate and the actual rate of the best effort TCP traffic without the gap control algorithm.

Gap Control Parameter (g) Analysis

We use 4 TCP connections to see how the SWAN AIMD rate controller controls TCP traffic. In this simulation, all TCP flows are greedy FTP type of traffic with packet size of 512 bytes. Figure 2.7 shows a trace of a TCP traffic flow that exhibits some inefficiency under our rate control regime. Note, that the actual and shaping rates do not match, and as shown in the trace, the shaping rate keeps growing while the actual rate of the TCP stalls and cannot follow the computed shaping rate. Such a disparity between the actual and shaping rates can be harmful. If left unresolved the "*gap*" can affect the performance of real-time traffic and other best-effort traffic. Such a condition arises naturally for TCP. For example, TCP traffic cannot follow the computed shaping rate when TCP traffic reaches the maximum throughput of the network or backs-off due to packet losses, etc. To resolve the mismatch of the actual and shaping rates, we introduce a gap control algorithm with parameter g, as described in Section 2.4.1. Figure 2.8 illustrates the trace of the TCP traffic with the gap

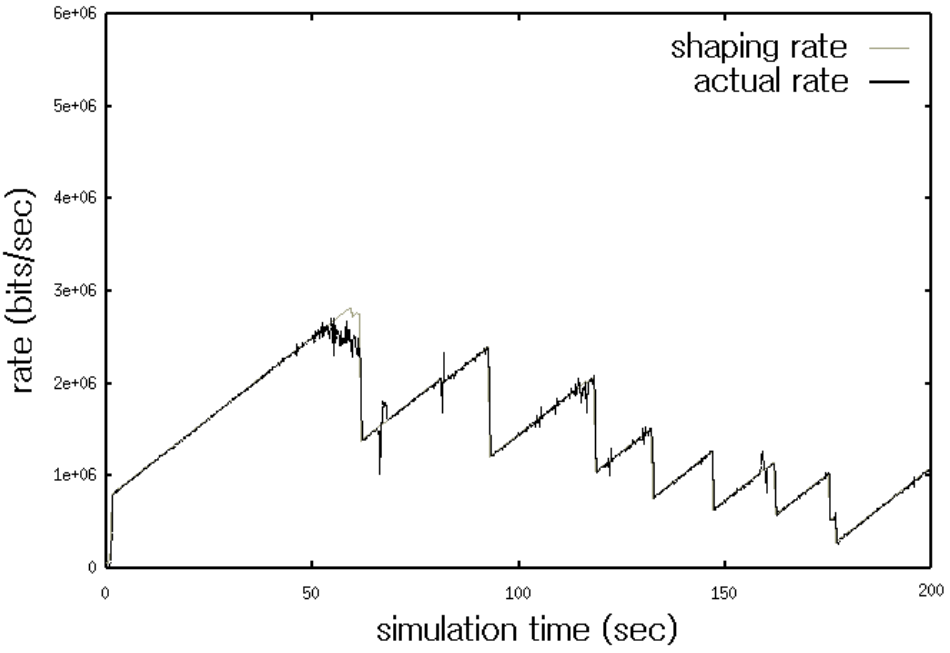


Figure 2.8: Trace of the shaping rate and the actual rate of the best effort TCP traffic with the gap control algorithm.

control algorithm activated. From Figure 2.8, we can observe that the actual rate closely follows the shaping rate providing better rate control for the TCP traffic; that is, gap control prevents a TCP from transmitting an uncontrolled, excessive burst of data, which could happen without gap control, as illustrated in Figure 2.7.

To understand the characteristic of parameter g, we measured the fairness between TCP flows (see Figure 2.9). The definition of fairness in [24] is used in measuring the fairness between TCP flows:

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)}, \tag{2.14}$$

where x_i is the fraction of the bandwidth allocated for i-th flow and n is the number of flows. F becomes 1 when all flows share the exactly same fraction. The x-axis in Figure 2.9 represents the value for parameter g (gap between actual rate and shaping rate, %). As shown in Figure 9, fairness tends to decrease as the value of g increases except when the value of g is less than 10%. A mobile device may have more chance of transmitting a burst

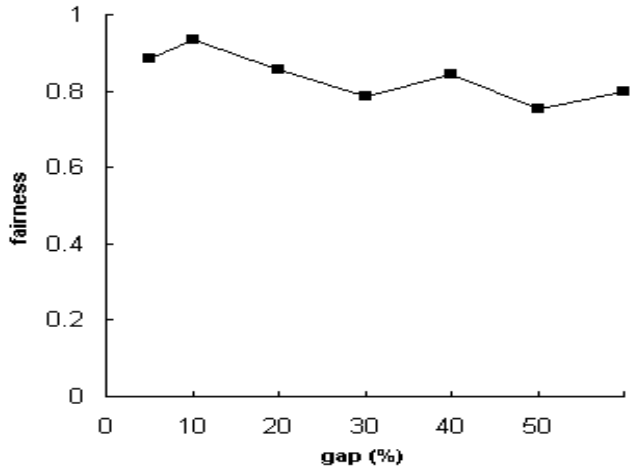


Figure 2.9: Fairness index of TCP traffic versus the gap control parameter.

as the value of parameter g increases. If a mobile device is allowed to transmit a burst then this burst may limit the performance of other TCP flows. As a result, the fairness decreases as the value of g increases. If the value of g is too small, the gap between the actual rate and the shaping rate will be greater than $g\%$ most of the time, and the gap control algorithm will set the shaping rate to closely match the actual rate rather than increasing the shaping rate with increment rate c. So the additive increase of the AIMD algorithm may not be able to operate efficiently. Thus, TCP flows with smaller traffic rates may have little chance to grow their rate with the result that fairness may eventually decrease.

AIMD Parameter (c, r) Analysis

To better understand the properties of the SWAN AIMD rate control parameters c and r, we consider two scenarios for background TCP best-effort traffic. The first scenario has eight TCP flows and the second has 32 TCP flows. In both scenarios, all TCP flows are greedy FTP type of traffic with packet size of 512 bytes. TCP flows are rate controlled with parameter c and parameter r, while voice and video flows are not rate controlled once admitted through the sourcebased admission control process. During the simulation, four voice and four video flows are active and monitored for the duration of 200 seconds representing real-time traffic.

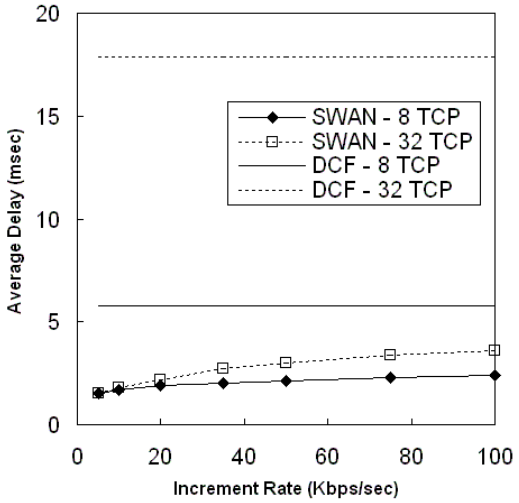


Figure 2.10: Average delay of real-time traffic versus increment rate.

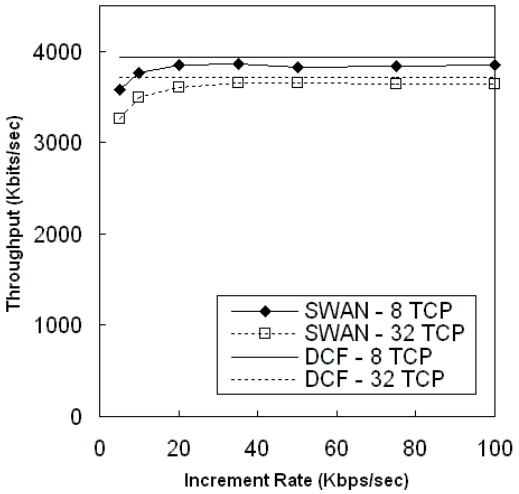


Figure 2.11: Total throughput of best-effort TCP traffic versus increment rate.

Voice traffic is modeled as 32 Kbps constant rate traffic with a packet size of 80 bytes. Video traffic is modeled as 200 Kbps constant rate traffic with a packet size of 512 bytes.

We measured the average MAC delay of real-time traffic (see Figures 2.10 and 2.12) and the total throughput of best-effort traffic (see Figures 2.11 and 2.13). The x-axis of Figures 2.10 and 2.11 represents the value for parameter c (increment rate, Kbps/ sec). The x-axis in Figures 2.12 and 2.13 represents the value for parameter *r* (decrement rate, percent). It is shown in Figure 2.10 that the value of parameter *c* does not have much impact on the average delay of real-time traffic. The average delay grows very slowly with the increasing

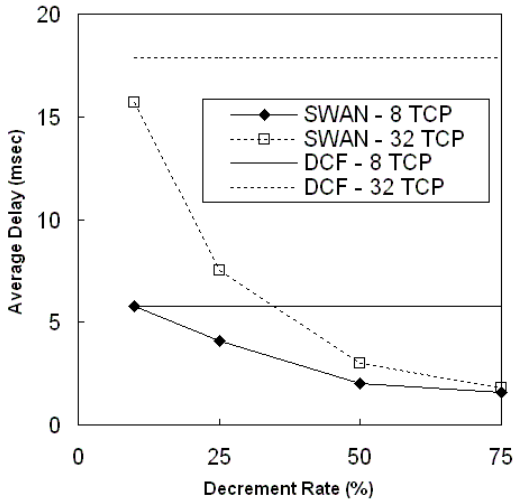


Figure 2.12: Average delay of real-time traffic versus decrement rate.

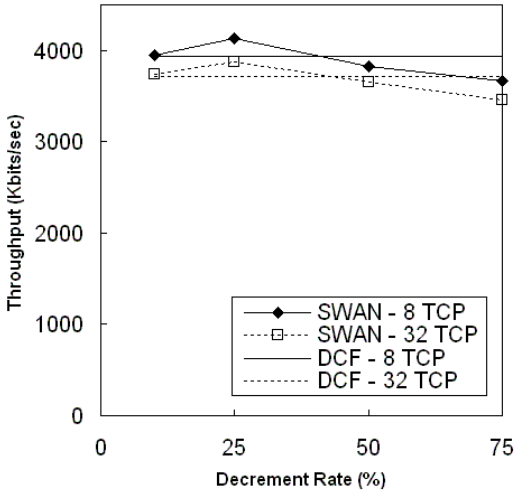


Figure 2.13: Total throughput of best-effort TCP traffic versus decrement rate.

value of parameter c. In contrast, the total throughput of best-effort TCP traffic is noticeably decreased when a small value of parameter c is chosen, as shown in Figure 2.11. When the increment rate is 5 Kbps/sec, throughput is reduced by about 10 percent for the eight TCP flow scenario and by 13 percent for the 32 TCP flow scenario in comparison to DCF. For an increment rate of 20 Kbps/sec or larger, the TCP throughput becomes almost constant with less than 3 percent reduction in throughput. The throughput of real-time traffic is 99.5 percent of the offered load, (i.e., less than 0.5 percent packet loss), in all cases for Figures 2.10 to 2.16.

The value of parameter *r* has significant impact on the average delay of the real-time traffic, as shown in Figure 2.12.When the decrement rate is set to 10 percent, the average delay becomes almost as large as the average delay in DCF. The average delay becomes smaller as the value of parameter *r* increases. It is observed in Figure 2.13 that the total throughput of the best-effort TCP traffic is also sensitive to the value of parameter *r*. SWAN shows the best and worst-case performance in terms of the total throughput of best-effort TCP traffic when the value of parameter *r* is 25 percent and 75 percent, respectively. When the value of *c* is 35 and the value of *r* is 50, the average delay of the real-time traffic is reduced by more than 60 percent with eight background TCP flows, and by 75 percent with 32 background TCP flows. These results demonstrate that we can achieve a reduction of 60-75 percent in the average delay of real-time traffic with a 2 percent loss of TCPthroughput using theSWANAIMDrate control algorithm. This is a promising result.

Comparison of DCF, CW_{min}, and SWAN

We now evaluate and compare the performance of DCF, CW_{min}, and SWAN. Figures 2.14 and 2.15 show the average delay of real-time traffic and the total throughput of TCP best effort traffic with a growing number of TCP sources, respectively. TCP traffic represents a mixture of greedy FTP traffic with packet size of 512 bytes and bursty Web traffic modeled as short TCP file transfers with random file size and random silent period between transfers. The file size is driven from a Pareto distribution with a mean file size of 10 Kbytes and a shape parameter of 1.2. The length of the silent period between two transfers is also Pareto in distribution with the same shape parameter with ameanof 10 seconds. This creates a highly bursty background best-effort traffic load over multiple time-scales. Web traffic represents microflows, whereas FTP traffic corresponds to macroflows. The realtime traffic is modeled in the same manner as discussed in the previous simulation using four voice flows of 32 Kbps and four video flows of 200 Kbps. In the CW_{min} simulation, a CW_{min} value

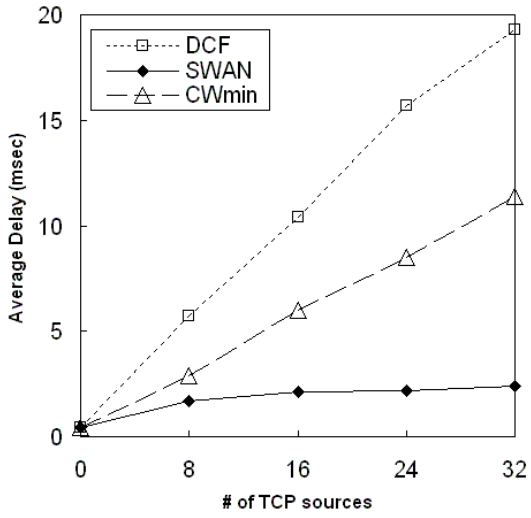


Figure 2.14: Average delay of real-time traffic versus number of TCP sources.

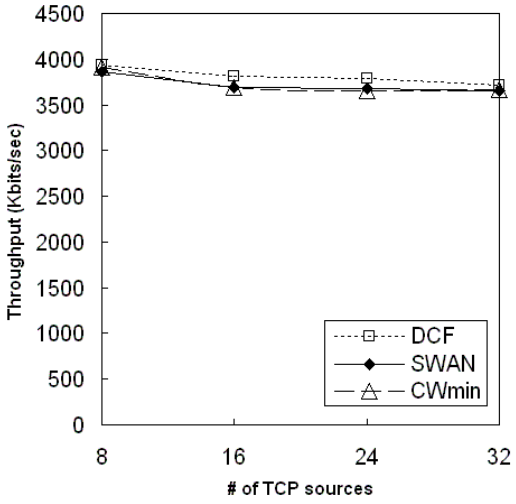


Figure 2.15: Total throughput of best effort TCP traffic versus number of TCP sources.

of 15 is used for real-time traffic and 31 for best effort traffic.

Figure 2.14 shows that the average delays for real-time traffic in all systems are the same without background TCP flows. The average delay of real-time traffic in DCF grows linearly from 5 to 19 msec when the number of background TCP traffic increases from eight to 32 flows. The average delay of real-time traffic in CW_{min} is improved over the case of DCF, but it also grows linearly from 3 to 11 msec when the number of background TCP traffic increases from 8 to 32 flows. In contrast, the average delay of real-time traffic in SWAN

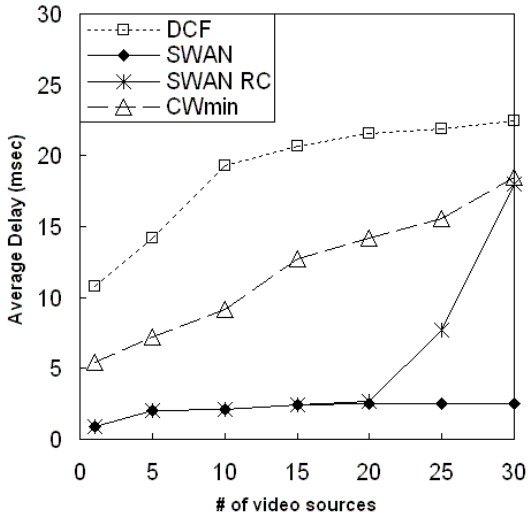


Figure 2.16: Average delay of real-time traffic versus number of video sources.

remains less than 3 msec. Figure 2.15 shows that the SWAN and CW_{min} systems lose only 2 percent of TCP throughput compared to DCF, while SWAN achieves up to 88 percent reduction in delay and CW_{min} only achieves up to 40 percent reduction.

Figure 2.16 shows the average delay of real-time traffic for a growing number of UDP video sources. The background TCP best effort traffic consists of 16 FTP and Web sources. In Figure 2.16, SWAN-RC refers to IEEE 802.11 DCF wireless networks with SWAN rate control (excluding SWAN admission control and regulation). DCF shows delays larger than 10 msec with only one video source and over 20 msec with 15 or more video sources. CW_{min} shows delays larger than 5 msec with only one video source and over 18 msec with 32 video sources. The SWAN-RC and SWAN schemes show the same performance with up to 20 video sources but SWAN-RC shows larger delays when there are more than 20 video sources. This result shows the necessity for SWAN admission control and regulation of real-time traffic. The results presented in this section show that wireless ad hoc networks with SWAN can support real-time traffic with consistently low delays in a single shared media channel. In the next section, we investigate the performance of SWAN that considers multihops and varying host mobility.

2.6.2 Performance of Multihop Scenarios with Mobility

In this section, we consider a simulated multihop network with 50 mobile ad hoc nodes. The network area has a rectangular shape of 1500m x 300m that minimizes the effect of network partitioning. AODV [78] is used for routing in the simulated network. The real-time traffic is modeled as four voice and four video flows. The background TCP traffic is modeled as a mixture of FTP and Web traffic. Typically, flows traverse 2-5 hops (three hops on average) between source-destination pairs.

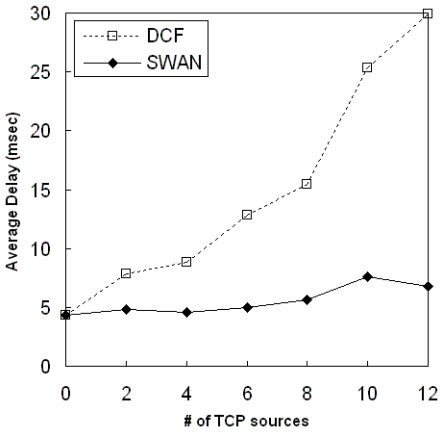


Figure 2.17: Average delay of real-time traffic versus number of TCP flows.

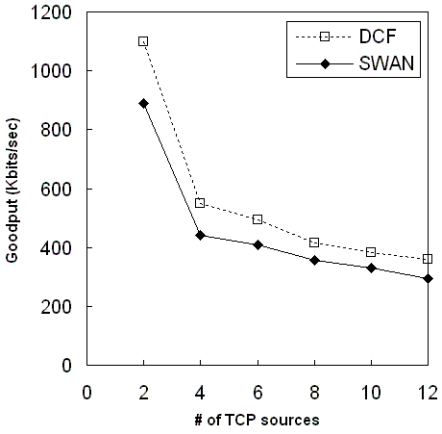


Figure 2.18: Average "goodput" of TCP best-effort traffic versus number of TCP flows.

Figures 2.17 and 2.18 show the average end-to-end delay for real-time traffic and the average goodput of the TCP best effort traffic for an increasing amount of background TCP traffic, respectively. The packet loss of the real-time traffic is less than 1 percent in both

DCF and SWAN. However, the average delay of the real-time traffic shows a significant difference between DCF and SWAN. The average end-to-end delay of the real-time traffic in DCF grows linearly from 8 to 30 msec as the number of TCP flows increase from 2 to 12 flows, respectively. In contrast, the average delay of real-time traffic in SWAN remains around 5 to 7 msec. The average goodput of the TCP traffic in SWAN is approximately 15-20 percent less than DCF. By adopting SWAN, we observe a 38-77 percent reduction in the average delay of the real-time traffic at a cost of 15- 20 percent loss of TCP goodput. In addition, the average delay of the real-time traffic remains consistently below 8 msec in SWAN while the average delay in DCF grows above 30 msec.

The impact of mobility is illustrated in Figures 2.19 and 2.20. The simulated network is the same as the previous multihop scenarios with the addition of the introduction of mobility. We use a random waypoint mobility model [13]. Each mobile node selects a random destination and moves with a random speed up to a maximum speed of 72 km/hr, pausing for a given pause time when the destination is reached. When the pause timer expires, the mobile node picks another random destination and moves at another random speed. The real-time traffic is modeled in the same manner as discussed previously. The number of best-effort TCP flows comprises five FTPs and five Web microflows. As shown in Figure 2.19, the average end-to-end delay of the real-time traffic in DCF increases slowly as mobility increases, and the average end-to-end delay of the realtime traffic in SWAN grows only for the highest mobility scenarios. We observed from the simulation results that the throughput of the real-time traffic decreases slowly from 99 percent to 95 percent of the offered load, (i.e., the packet loss increases from 1 to 5 percent), as mobility increases in both DCF and SWAN. The impact of mobility on delay and throughput is due to route discovery latency and congestion along the new route. However, the end-to-end average delay of the real-time traffic in SWAN remains under 10 msec in all cases, while the average delay in DCF grows to 38 msec. The average goodput of best-effort TCP traffic in SWAN is about 15-25 percent less than DCF, as shown in Figure 2.20. In SWAN, the average

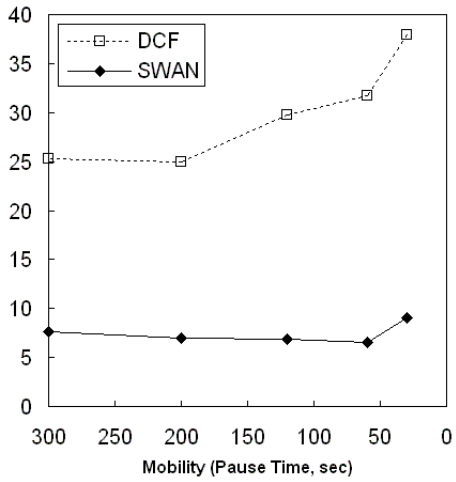


Figure 2.19: Average delay of the real-time traffic versus mobility.

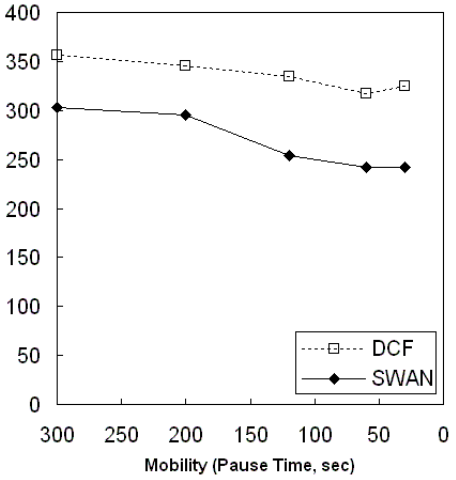


Figure 2.20: Average goodput of the best-effort TCP traffic versus mobility.

end-to-end delay of the real-time traffic is reduced by 70-75 percent with 15-25 percent loss of best-effort TCP goodput. The average end-to-end delay of the real-time traffic in SWAN stays consistently below 10 msec while the average delay in DCF grows to 38 msec.

2.7 Wireless Testbed Results

In what follows, we describe our experimental results from the SWAN wireless testbed, which is based on Linux notebooks using Aironet IEEE 802.11b wireless interfaces. The rate controller is implemented by modifying the Aironet device driver. We also modified

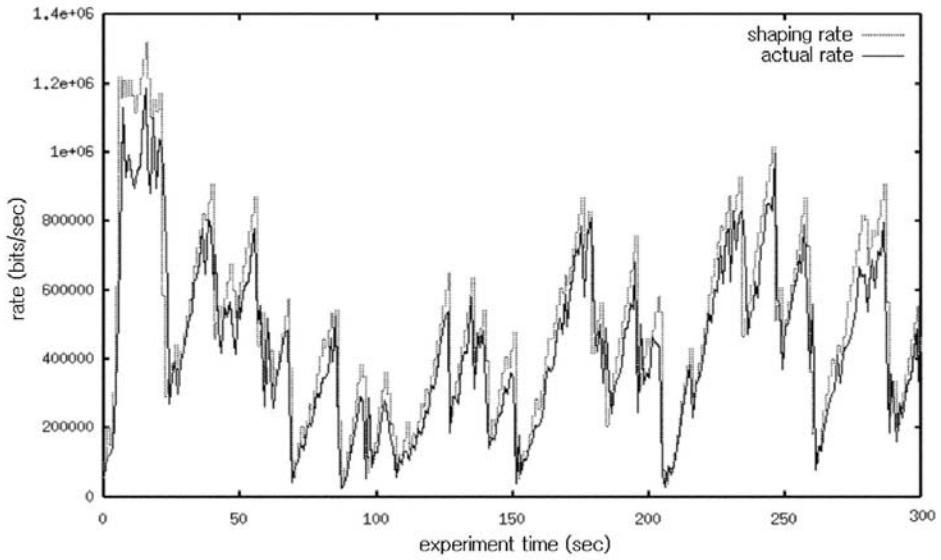


Figure 2.21: Trace of the shaping rate and the actual TCP transmisson rate.

the host driver to measure packet delay. The packet delay is measured by calculating the difference between the time the device driver feeds a new packet into an Aironet card and the time the Aironet card acknowledges back to the device driver that the transmission of the packet is successful. A Linux-based traffic shaper operating between the kernel and the Aironet card device driver is used to control the rate of TCP traffic. The utilization monitor and probe protocol are implemented using the Berkeley Packet Filters Packet Capture library (PCAP). PCAP is designed to capture packets for statistical purposes but it can also be used to forward packets to the network interface. PCAP is used to capture every UDP packet transmitted within the radio coverage range of wireless mobile hosts. The admission controller reads the IP header of captured real-time UDP packets and estimates the local bandwidth availability. We used AODV to find a route from the source to the destination for UDP, TCP, and signaling packets. The admission controller estimates the end-to-end bandwidth availability when a source node probes the network path, as discussed previously. SWAN control algorithms are implemented as a separate daemon from the IP forwarding engine (inside the kernel) and the AODV routing daemon.

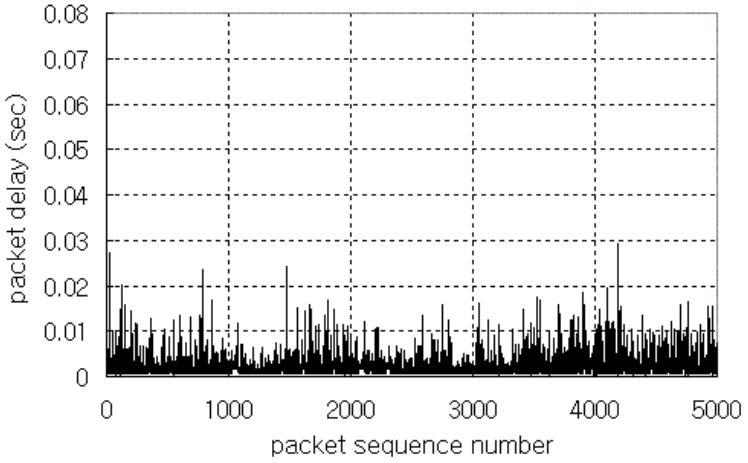


Figure 2.22: The delay of each packet in a UDP real-time flow from the wireless testbed with SWAN.

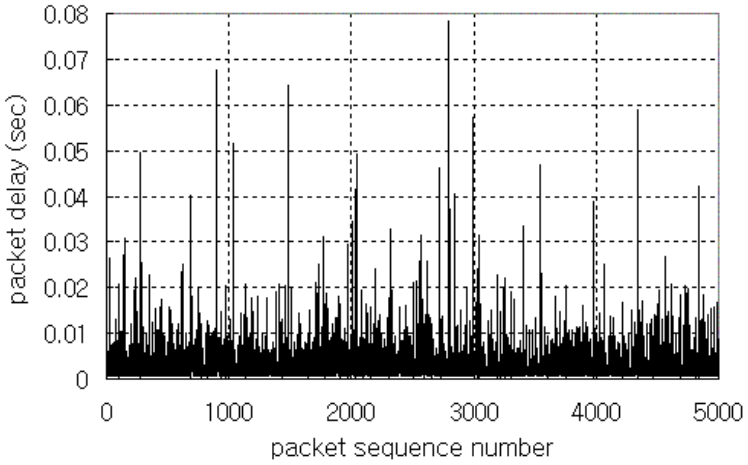


Figure 2.23: Delay of each packet in a UDP real-time flow from the wireless testbed without SWAN (pure DCF).

The results presented in this section were obtained from an experimental SWAN wireless ad hoc testbed, which consists of five mobile hosts using Aironet 11 Mbps IEEE 802.11b PCMCIA cards. The configuration of the testbed is as follows: Four mobile hosts generate TCP traffic and one mobile host generates UDP traffic. The source and the destination nodes associated with each flow are distributed among the mobile hosts. All mobile hosts share a single media channel. The UDP host generates packets every controlled by the rate controller. The TCP traffic is a mixture of FTP and Web flows, and, the UDP traffic

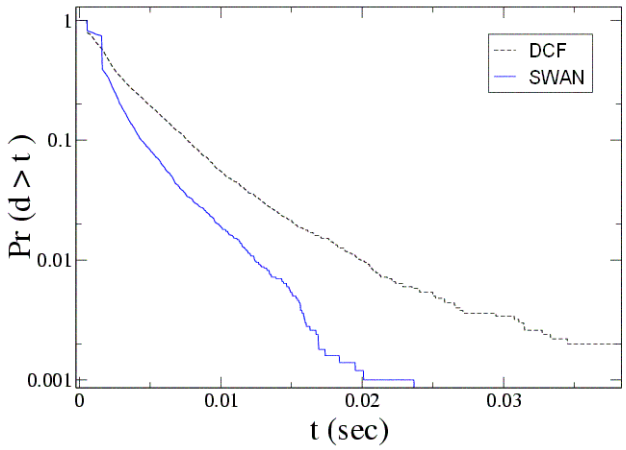


Figure 2.24: The normalized distribution of the delay of the packets in a real-time traffic.

represents MPEG-1 video flows.

Figure 2.21 shows a traffic trace of the shaping rate controlled by the rate controller and the actual TCP transmission rate. The actual TCP rate is well controlled by the shaper, as shown in the figure. When all four TCP flows are rate controlled, we measure the delay of each packet in a UDP real-time flow. Figures 2.22 and 2.23 show the delay of each packet when the TCP flows are regulated and unregulated, respectively. By comparing Figures 2.22 and 2.23, we can observe that the measured delay is improved when TCP flows are rate controlled. The average measured delay is 2.3 msec and 3.3 msec in Figures 2.22 and 2.23, respectively. The average delay difference between the results shown in the figures is not significantly large because the number of TCP mobile nodes in the testbed is small (i.e., only four nodes). The average delay observed in Figures 2.22 and 2.23 matches the average delay of the four TCP mobile nodes shown in Figure 2.14. Even though the average delay difference is not significant,wecan still observe that the measured delay shown in Figure 2.22 remains below a certain boundary most of the time, while the delay shown in Figure 2.23 frequently reaches significantly higher values.

Figure 2.24 shows a normalized distribution of the measured UDP real-time packet delay for DCF, (i.e., the wireless testbed without SWAN, as shown in Figure 2.23), and SWAN

(i.e., the wireless testbed with the SWAN algorithms implemented, as shown in Figure 2.22). Compared to DCF, SWAN is less likely to exceed t seconds, where t is greater than 2 msec. We can observe the contrast more clearly from the tail of the distribution shown in Figure 2.24. This result shows that SWAN improves the performance of real-time applications in terms of delay distribution.

2.8 Conclusion

In this chapter, we proposed SWAN, a simple, distributed, and stateless network model that uses distributed control algorithms to support real-time applications and service differentiation in mobile wireless ad hoc networks. An important benefit of SWAN is that it is independent of the underlying MAC layer, and can be potentially suited to a class of physical/data link wireless standards. We presented the performance evaluation of SWAN using the ns-2 simulator, and analyzed the MAC delay and busy probabilities, confirming SWANs design decisions. We compared the performance of DCF, CWmin, and SWAN using analysis and simulation. The results show that DCF requires SWAN rate control, admission control, and regulation to support real-time traffic. Simulation, analysis, and results from our experimental wireless testbed show that real-time applications experience low and stable delays under various multihop, traffic, and mobility conditions with SWAN. The SWAN testbed and ns-2 source code are available from the Web [98]. Finally, an IETF Internet Draft describes the full SWAN specification [2].

Chapter 3

Funneling-MAC

3.1 Introduction

Wireless sensor networks exhibit a unique funneling effect [96] where events generated in the sensor field travel hop-by-hop in a many-to-one traffic pattern toward one or more sink points, as illustrated in Figure 3.1. This combination of hop-by-hop communications and centralized data collection at a sink creates a choke point on the free flow of events out of the sensor network. For example, the funneling of events leads to increased transit traffic intensity and delay as events move closer toward the sink, resulting in significant packet collision, congestion, and loss; at best this leads to limited application fidelity measured at the sink, and at worst the congestion collapse [49] of the sensor network. Other drawbacks exist. The sensors nearest to the sink, typically within a small number of hops loose a disproportionate larger number of packets (we call this region of the funnel the intensity region, as illustrated in Figure 3.1) and consume significantly more energy than sensors further away from the sink, hence, shortening the operational lifetime of the overall network. Mitigating the funneling effect represents an important challenge to the sensor network community and is the subject of this chapter.

Researchers have proposed distributed congestion control algorithms [49], tiered network

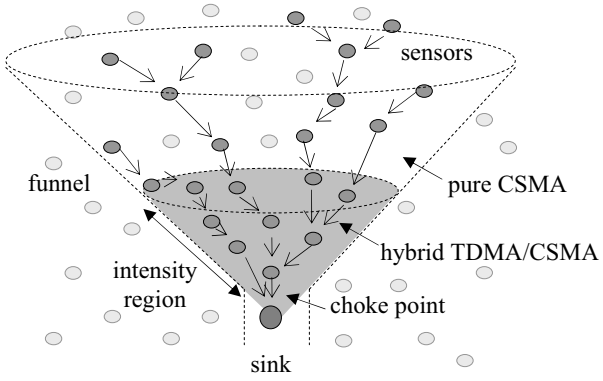


Figure 3.1: Funneling effect in sensor networks.

design [96], and data aggregation techniques [79] [91] to respond to increased load and congestion in sensor networks. But as the literature [49] [96] indicates these techniques alone cannot fully alleviate the problem because it is very difficult to effectively rate control traffic at aggregation points or sources to match the bottleneck conditions observed at the sink nodes. In this chapter, we show that the majority of packet loss in a sensor network occurs within the first few or more hops from the sink, even under light traffic conditions. We conjecture that by putting additional control within the first few or more hops from the sink we can significantly improve communication performance and eradicate the funneling effect.

We propose a localized, sink-oriented funneling-MAC that explicitly recognizes the existence of funneling effect in its design. While there have been a number of important new MAC protocols proposed for sensor networks, to the best of our knowledge none have addressed the funneling effect. The funneling-MAC represents a hybrid (schedule-based) TDMA and (contention-based) CSMA/CA MAC scheme that operates in the intensity region of the event funnel, as illustrated in Figure 3.1. Pure CSMA/CA operates network-wide in addition to acting as a component of the funneling-MAC that operates in the intensity region. The funneling-MAC mitigates the funneling effect by using local TDMA scheduling in the intensity region only, providing additional scheduling opportunities to nodes closer to the

sink, which typically carry considerably more traffic than nodes further away from the sink. The funneling-MAC is sink-oriented because the burden of managing TDMA scheduling of sensor events in the intensity region falls on the sink node, and not on resource limited sensor nodes. The funneling-MAC is localized in operation because TDMA only operates in the intensity region close to the sink and not across the complete sensor field. The burden of computing and maintaining the depth of the intensity region also falls on the sink. We assume that the sink is likely to have more computational capability and energy reserves than simple sensors; however, the funneling-MAC does not rely on this to operate efficiently. By using TDMA in this localized manner, and putting more management onus on the sink not the sensors, we offer a scalable solution for the deployment of TDMA scheduling in sensor networks, one that is capable of boosting application fidelity as measured at the sink, but does not have the scalability problems associated with the network-wide deployment of TDMA, which, we believe, is untenable today as a network-wide deployment strategy for large-scale sensor networks.

The structure of the chapter is as follows. In Section 3.2 we show the impact of the funneling effect using results from an experimental sensor network. The effectiveness of existing MACs to counter the funneling effect is discussed in Section 3.3. Following this, we present the detailed design of the funneling-MAC algorithms in Section 3.4 that include: on-demand beaconing, which both provides light-weight clock synchronization for TDMA scheduling in the intensity region, and regulates effectively boundary of that region; sink-oriented scheduling, which computes and distributes new schedules when needed in an efficient low cost manner; and dynamic depth-tuning, which dynamically adjusts the depth of TDMA operating in the intensity region with the goal of maximizing the throughput of the sink choke point while minimizing the packet loss in the funnel. Section 3.5 provides an important analytical foundation that justifies the choice of dynamically controlling the depth of the intensity region in response to measured traffic conditions at the sink node. Section 3.6 presents results from a number of experiments from various setup using up to 45 mica-2

motes. We take an experimental systems approach for the validation of the funneling-MACs performance. We consider a number of different node densities, and traffic characteristics to study the performance of the funneling-MAC in comparison to other representative protocols such as the TinyOS [100] default protocol B-MAC [80], and more recently proposed, and comparative protocol Z-MAC [87], which is also based on a hybrid TDMA/CSMA approach. We show by simply exerting control over the first few or more hops from the sink that the funneling-MAC significantly outperforms B-MAC and Z-MAC, which we show are not capable of dealing with the funneling effect.

The funneling-MAC source code is freely available from the project webpage [97] and TinyOS Source Forge [100].

3.2 Funneling Problem

We begin by first quantifying the impact of the funneling effect in a sensor network using the TinyOS CSMA-based B-MAC protocol, the MintRoute routing protocol, and the Surge application in a 45 mica-2 testbed. The network is deployed as a 5x9 rectangular grid of equally spaced motes in a large open room, making sure there are no interference and near-field issues [22] during the experiments. The mote at the bottom left corner operates as

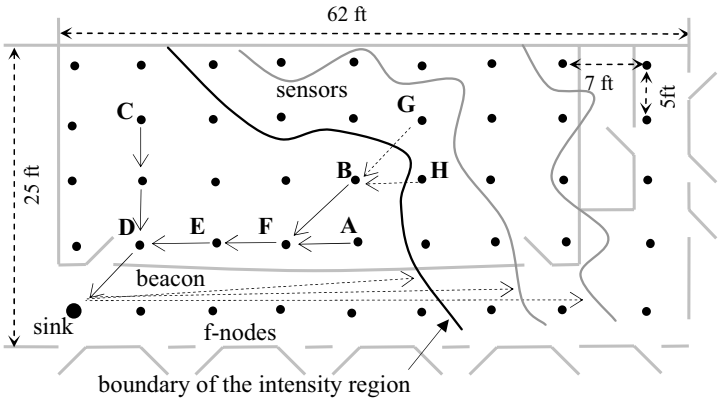


Figure 3.2: Dartmouth College sensor testbed.

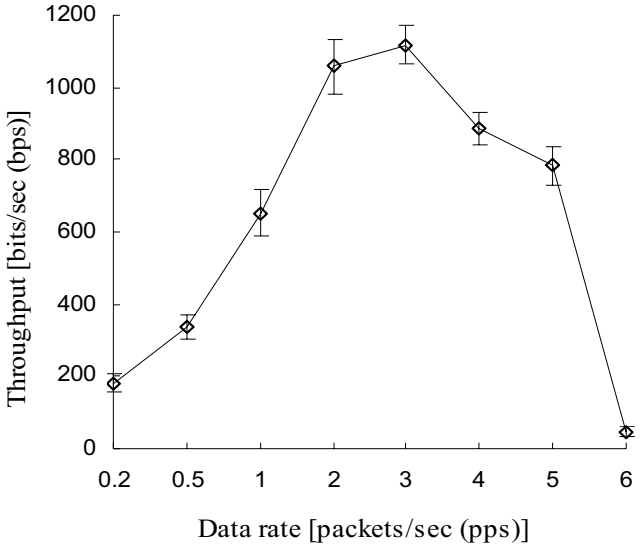


Figure 3.3: Throughput of CSMA with varying data rates.

the sink in the grid, as illustrated in Figure 3.2. Node spacing and transmission power are set such that one-hop neighbors achieve >80% delivery, while two-hop neighbors achieve <20% delivery. In this way, a fairly strict and dense multi-hop radio environment is constructed for experimentation.

We randomly select 16 of the 44 sensing nodes to generate event rates ranging from 0.2-5 packets/sec (pps) where the packet size is 36 bytes. The goal is to gradually drive the sensor network from low to moderate load and then into a congested and saturated state, while studying the choke point throughput measured at the sink and the loss in the network. Typically, events travel over multiple hops, 2-5 hops in the case of the experiment. Figure 3.3 shows the resulting fidelity (i.e., throughput curve), as measured at the sink as we increase the event rate of all 16 sources. Note that we exclude the preamble and CRC sizes, and count the packet size as 36 bytes when calculating the throughput fidelity. We can clearly see that the throughput measured at the sink rises to a peak of approximately 1100 bps before the network falls into a congested and saturated state. Further increase in source rate only drives the network into further overload and eventual collapse with increasing load.

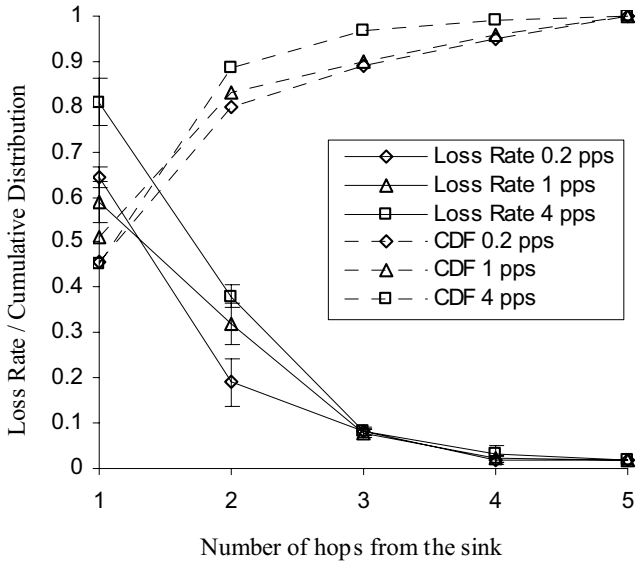


Figure 3.4: Loss rate and cumulative distribution function of loss loss over varying distance from the sink for CSMA.

We observe from Figure 3.3 that source rates of 0.2 pps, 1 pps, and 4 pps can be considered to be light, medium (near optimal load), and overload traffic scenarios, respectively. We use these rates to further study the impact of the funneling effect on loss distributions across the network. We consider the overall loss rate in the network to be the number of packets lost in the network divided by the number of packets transmitted in the network. The overall loss rates measured for increasing load are approximately 67%, 72%, and 95% loss rate for 0.2 pps, 1 pps, and 4 pps, respectively. What is surprising about these results in that at low load there is still significant loss (67%), which rises to the point where 95% of events transmitted in the network are lost at high load. This also translates to significant energy waste. Such loss is unacceptable for many applications and would quickly deplete the sensors energy reserves. Note that in the case of light and medium traffic scenarios, packet loss is mainly due to collision and hidden terminal problem, whereas in the high and overloaded traffic scenarios loss is due to buffer overflow in addition to collision and hidden terminal problem.

Next, we investigate where the losses occur in the network. The solid lines in Figure 3.4 show the loss rate at the i-th hop (i.e., the ratio of the lost packets among the packets that are

transmitted by the nodes at the i-th hop). This result clearly quantifies the funneling effect for this experiment and shows its debilitating impact on network performance. These results represent the average of five runs of the same experiment and the 95% confidence intervals. What is interesting about these results is that Figure 3.4 clearly shows that there is increasing loss at nodes closer to the sink, which is a product of the many-to-one, hop-by-hop traffic pattern of the funneling effect. For example, for all traffic rates the vast majority of packet loss occurs in the first two hops from the sink and the loss rate drops quickly for hops further away from the sink. These are fingerprints of the funneling effect. Note, that even for a light traffic load of 0.2 pps this trend is still dominant with significant loss registered in the first few hops. These per-hop loss rates for the low rate traffic explain why at such a low rate we still can record an overall loss rate for the network of 67%, as discussed above. The dotted lines in Figure 3.4 show a cumulative distribution function (CDF) of the per-hop losses. We can observe from the plot that between approximately 80-90% of the losses across the three low, medium, high rates happened within the first two hops from the sink. We can conclude that funneling effect is mostly invariant to source rate.

These results indicate that additional controls (e.g., scheduling) in the network over the first few hops could offer significant gains across all traffic rates considered in the experiment (viz. light, medium, heavy). We can also conclude that even at low rates the CSMA-based B-MAC cannot mitigate the funneling effect. These are important insights. Therefore, we conjecture that new MAC approaches other than B-MAC are needed to fully address the funneling problem.

3.3 Related Work

In what follows, we discuss a number of sensor network MAC protocols and traffic control mechanisms found in the literature and comment on how they would perform in mitigating the funneling effect discussed in the previous section. S-MAC [106], T-MAC [28], B-MAC

[80] and the MAC discussed in [101] represent well-known contention-based (CSMA) MAC protocols for sensor networks. In [101] the authors discuss an early contribution to sensor network MACs that uses adaptive rate control mechanisms on top of CSMA to achieve energy efficiency and fairness. This MAC [101] represents a network-aware scheme like the funneling-MAC in the sense that it considers route-through traffic when using rate control. S-MAC avoids idle listening by putting sensor nodes to sleep periodically. S-MAC requires time synchronization but the time-scale is much larger than TDMA. T-MAC provides almost the same functionality as S-MAC except that it is capable of further reducing the idle listening by transmitting all messages in the buffer of each node at the beginning of the active period, allowing it to sleep instantly once the buffer is flushed. B-MAC provides well-defined interfaces to low power listening (LPL), clear channel assessment (CCA) and acknowledgements. LPL improves the energy efficiency and throughput with the cost of transmitting a long preamble by sources. We show that B-MAC is not capable of mitigating the funneling effect because of the large build up of losses in nodes closer to the sink, as discussed in the previous section. WiseMAC [34] also use the preamble technique to support low power listening. In WiseMAC, each node learns the wakeup schedule of its neighboring node and optimizes the preamble size. Researchers have also proposed mechanisms to minimize the idle listening to further reduce energy consumption by embedding the destination node information in the packet preamble and strobing the preamble (X-MAC) [14] or synchronizing the channel sampling slots to reduce the preamble size (SCP-MAC) [107]. RI-MAC [94] uses receiver-initiated data transmission to eliminate the reliance of long preamble. A contention based protocol named E-CSMA [33] has been proposed to adopt a learning approach at the transmitter to estimate the probability of successful packets reception at the receiver by exploiting low cost channel feedback. We conjecture that all those MAC protocols we have mentioned above are based on similar contention-based approaches which would likely be as non-responsive and show the same poor trends as B-MAC in dealing with the funneling effect.

There are several schedule-based (TDMA) MAC algorithms proposed in the sensor network literature that do better at mitigating the funneling effect. The energy-aware TDMA-based MAC [5] achieves collision free access and energy efficiency by assigning each node their own time slots (listening slot and transmitting slot), allowing nodes to sleep when it is not their slot time. This approach [5] may be impractical because the sink requires complete topology information to compute the TDMA schedule and every node requires precise time synchronization. Furthermore, in [5] the schedule is delivered to the nodes in a hop-by-hop fashion which implies that, if a schedule packet is lost, the cost for recovering the missing schedule might be too high. These issues indicate that the actual implementation of such a scheme in a large sensor network would have scalability problems.

Another TDMA protocol called TRAMA [83] performs an adaptive election algorithm to overcome this drawback of wasting time slots. TRAMA is a scalable distributed algorithm where each node schedules time slots among its two hop neighbors using a neighbor protocol and schedule exchange protocol as discussed in [83]. One drawback of implementing TRAMA in a mote network (no current implementation exists for TinyOS, as far as we are aware) is that the overall signaling overhead of these fairly complicated protocols may present scalability problems, particularly if implemented in a large-scale testbed. FLAMA [82] is built upon TRAMA to remove the periodic traffic information exchange in the two hop neighborhood which is instead now operated upon request when an application flow is established. A protocol called D-MAC [66] adopts staggered TDMA schedule to schedule transmissions/receptions along a path from a source to the sink to avoid the data-forwarding interruption problem present in duty cycled protocols like S-MAC. D-MAC does not address the funneling effect because it assigns only one slot per node. There are a number of other TDMA-based algorithms found in the literature [77] [38] [63] (but not implemented in mote networks) that suffer from similar problems when targeted toward large-scale sensor deployment because of the need for global network-wide schedule computation and distribution, and time synchronization.

The most suitable protocol for potentially mitigating the funneling effect that is available in source code for mica-2 motes is the Z-MAC protocol. Z-MAC [87] is a hybrid protocol that acts like a contention-based protocol under low traffic conditions and a schedule-based protocol under high traffic conditions by using the schedule computed by DRAND [88] as a hint. DRAND is a distributed implementation of RAND [86], which is a centralized channel reuse scheduling algorithm. RAND is not scalable solution because it requires the topology information of the entire network. DRAND requires the topology information of two hop neighbors. DRAND allocates time slots to every node ensuring that no two nodes among a two-hop neighborhood are assigned to the same time slot by broadcasting the TDMA schedule of each node to its two hop neighbors.

Z-MAC reduces the hidden terminal problem by not allowing two nodes in two-hop distance to transmit at the same time. In order to improve utilization, Z-MAC allows non-owners of a slot to contend for the slot if it is not being used by its owner. Z-MAC requires global time-synchronization in the initial phase, and then it performs local synchronization by sending periodic sync packets between nodes. Z-MAC requires that DRAND is run at startup to set up the TDMA schedule, which may be a heavy burden for light-weight sensor devices. The message complexity of DRAND is O(δ), where δ is the local neighborhood size of each node while the message complexity of the funneling-MAC (detailed in the next section) is O(1). Because of the overhead of running DRAND, the Z-MAC authors do not recommend that it be run periodically. We choose to compare the funneling-MAC to Z-MAC in the experimental evaluation section (Section 3.6). We note in those experiments that Z-MAC is susceptible to schedule drift (i.e., when the schedule allocated by DRAND to nodes drifts out of sync because of various time varying radio impairments). We discuss these issues and show that, while Z-MAC offers scheduling support, it is not designed to schedule more traffic at nodes closer to the sink in its current form, and therefore, cannot mitigate the effects of funneling events to a sink choke point. Because of the potential for schedule drift, Z-MACs performance ends up degrading to being only marginal better than

B-MAC under a number of experimental scenarios, as we discuss in Section 3.6.

Crankshaft [42] is another hybrid approach that use simpler scheduling than DRAND. Crankshaft assigns a receiving time slot for each node by calculating MAC address modulo n, where n is the number of slots in a frame. In each slot, nodes that have some packet to send to the receiving node of the slot contend to transmit. In a time slot, only the receiving node and the contending nodes are awake. The sink nodes listen to all slots because the sink is the destination for most traffic and the sink typically has more energy than sensor nodes. The special treatment for the sink potentially mitigates the funneling-effect to some extent. However, Crankshaft has limitation in mitigating the funneling-effect because the contention in Crankshaft is higher than the contention in Z-MAC (not to mention the funneling-MAC). The reason is that Crankshaft does not have the designated sending node of a slot. In addition, nodes that are transmitting to different receiving nodes may also contend because Crankshaft does not prevent two neighboring nodes to be assigned the same slot.

Flexible Power Scheduling (FPS) [46] also represents a hybrid approach that provides coarse grain scheduling that computes radio on/off times, and fine grain MAC control for channel access. The coarse grain scheduling of FPS represents a distributed approach where each node schedules its own children. The funneling-MAC and Z-MAC have some similarities to FPS. However, FPS is limited when dealing with the funneling effect because it does not prevent nodes with different parents from using the same slot. FPS simply relies on CSMA to provide collision avoidance in this case.

In [96] the authors propose to add multi-radio virtual sinks to sensor networks as a means of dealing with loss at the physical sink. Virtual sinks address the funneling effect by adding more capacity in an on-demand manner to the network using network layer routing to redirect traffic off the primary mote radio network (reducing the funneling effect on the physical sink) and onto an overlay network. While virtual sinks are effective they require specialized multi-radio nodes and an overlay network to siphon packets off the primary network. In addition, virtual sinks themselves can experience a mini-funneling effect [96].

3.4 Funneling-MAC Design

We now discuss the detail design of the funneling-MAC algorithms, and issues related to timing and framing.

3.4.1 On-Demand Beaconing

The funneling-MAC localized TDMA is triggered by a beacon broadcasted by the sink. All sensor nodes perform CSMA by default unless they receive a beacon and are then deemed "f-nodes". The sink regulates the boundary of the intensity area (see Figure 3.2) by controlling the transmission power of the beacon. The dynamic depth-tuning algorithm discussed in Section 3.4.5 determines this transmission power. The sink then transmits the beacon message at the computed transmission power. The nodes that received the beacon consider themselves to be in the intensity region and f-nodes. These nodes can perform TDMA while the nodes that do not receive the beacon (e.g., those nodes outside the intensity region) perform CSMA. F-nodes need to synchronize their clock to perform TDMA but the funneling-MAC does not rely on any synchronization protocol. If a network synchronization protocol is present then the funneling-MAC can use that and further minimize its active beacon signaling. However, in our implementation of the funneling-MAC we do not assume this and integrate a light-weight clock synchronization scheme embedded in the beacon messaging. Therefore, f-nodes rely on the beacon sent to activate TDMA and regulate the boundary of the intensity region for clock synchronization. As soon as a node receives a beacon, it becomes an f-node and synchronizes with other f-nodes by initializing its clock. The propagation delay of a beacon is on the scale of microseconds in wireless sensor networks while the accuracy of synchronization required for the funneling-MAC is on the scale of milliseconds, so beacon-based synchronization can keep the synchronization tight enough to perform TDMA scheduling. Because the beacon is broadcast across the complete intensity region then all f-nodes receive the beacon at the same time and are tightly

synchronized. This is a similar approach to reference-broadcast synchronization [35] but much simpler.

The beacon packet contains a small number of control fields including the beacon interval, superframe duration, and the TDMA duration. The superframe duration and TDMA duration are explained in Section 3.4.3 on framing. The beacon is sent periodically every beacon interval specified in the beacon packet. Experimentally we set the beacon interval so it is responsive to possible changes in routing, traffic rates, and clock drift of f-nodes. The beacon interval is determined by taking into account the accuracy of the local clock of the motes and required accuracy of the synchronization, as discussed in Section 3.6.1.

The beacon is sent only when it is necessary and in an on-demand basis. The beacon is not sent when the network is idle or receiving very low traffic. Note that every f-node keeps a timer that expires if the f-node does not receive a beacon for a period longer than the beacon interval. When the timer expires, the node performs pure CSMA. As soon as the sink receives a sufficient amount of data packets as determined by a change in the weighted moving average of the traffic (measured at the sink) from all paths then it begins to transmit a beacon periodically, based on the computed beacon interval. Conversely, if the sink does not receive sufficient traffic to allocate slots in the network in one or more beacon interval times, then it stops sending beacons until the sink registers such a positive change. F-nodes use the beacon interval to synchronize with future beacon transmissions from the sink. A mote based beacon interval timer allows motes to defer from transmitting when a beacon is due which would potentially interfere with the beacon if left unregulated.

When the sink starts beaconing at start-up or just after an idle period, it starts with the minimum transmission power (i.e., the same transmission power as ordinary sensor nodes). This is because the depth-tuning algorithm (as described in Section 3.4.5) uses an incremental increase/decrease rule when calculating the beacon/schedule transmission power. Gradually the sink will increase the transmission power as the measured traffic increases and the throughput/loss objectives are met (as discussed in the Section 3.5) using

the dynamic depth-tuning algorithm. Conversely, if the sink was to send the beacon not at the minimum power as discussed but rather high transmission power from start-up or after an idle period, then the beacon would likely interfere with contention based incoming CSMA data packets. This is because motes in a start-up state or just after an idle period are not aware when a beacon will be transmitted. This problem is resolved by the funneling-MAC because the starting point for the dynamic depth-tuning algorithm is always the same as the common default power used by motes (which is considered to be the power floor for the depth-tuning algorithm). Hence the impact of interference is minimized. Since the objective of the tuning algorithm is to increase the depth of the intensity region and therefore the transmission power there is a case that nodes not reachable by the existing power level will be interfered with when the tuning algorithm increments the beacon transmission power. The funneling-MAC resolves this potential interference issue by introducing a meta-schedule advertisement (which is discussed in Section 3.4.4).

Our design goal is to limit the cost of supporting periodic beacons by making them on-demand. One other parameter we consider is to extend the beacon interval to trade off signaling overhead, the reception power used by motes in the existing intensity region, and reduce the energy demands on the sink. We introduce the notion of "lazy beaconing" which pushes out the optimal beacon interval that is used to maintain tightness of clock synchronization and slot scheduling at f-nodes. By pushing out the beacon interval in this manner there can be some performance penalties if left unbounded. In Section 3.6.1, we discuss the optimal beacon interval used to maintain tight synchronization and slot scheduling, and optimal throughput, and contrast this to lazy beaconing which allows us to triple the optimal beacon interval for only a small reduction in the performance of the network, as measured by sink fidelity.

3.4.2 Sink-Oriented Scheduling

The sink monitors the traffic that arrives at the sink on a per-aggregated-path basis, calculates the TDMA schedule based on the monitored traffic (initially based on only new CSMA events and thereafter including existing TDMA traffic) for all paths, and distributes the schedule by broadcasting a schedule packet at the same transmission power used by beaconing. We define an aggregated path as a path which results from the merge of two or more paths at or before entering the intensity region. The funneling-MAC treats an aggregated path as a single path entry. For example in Figure 3.2, the funneling-MAC keeps information associated with paths G-B-F-E-D and H-B-F-E-D as a single aggregated path entry B-F-E-D. The funneling-MAC scales well because the number of aggregated paths entering the intensity region is bounded by the number of nodes in the intensity region. We use the term path to indicate aggregated path in the remainder of the chapter for convenience. In what follows, we provide a detailed discussion of sink-oriented scheduling.

In order to compute the schedule the sink needs to determine the identity of the path head f-nodes and the weighted average of the traffic on the path in order to correctly schedule the path. The concept of a path represents the direction taken by a train of events from a path head (e.g., mote A in Figure 2) on a hop-by-hop basis along a route (e.g., determined by the TinyOS MintRoute routing protocol in our experiments) to the sink (e.g., path A-F-E-D-Sink). The sink measures the weighted moving average of each path and allocates slots according to an allocation rule, which we discuss below. In order to enable the sink to acquire this information the funneling-MAC reserves 3 bytes in the packet header called the path information field. The path information field is only updated by the f-nodes along a certain path in the intensity region. The sink gathers this information from incoming packets on a per-path basis for all paths in the intensity region. The path information field contains the path head id (2 bytes) and the number of hops (1 byte). The path head lies near the intensity region boundary where the path head id equals the node id of the path head, and the

number of hops field reflects the number of hops the packet traverses on the path between the path head and the sink. For example in Figure 2 if a packet generated from outside of the intensity region is received by node A, node A forwards the event packet toward the sink following the path A-F-E-D-Sink. In this simple example, the path head id is A, and the value of number of hops is 4. Importantly, node A identifies itself as the path head when it receives a data event packet with a value of the path information field set to zero. In addition, source nodes inside the intensity region identify themselves as a path head when they generate a new packet. A path head puts its id in the path head id field and a value 1 in the number of hops field. All f-nodes along the path increment the value of the number of hops field by 1 when they forwards the event data packet. Consequently, each packet that arrives at the sink carries the path head id of the path it traversed as well as the number of hops.

The sink monitors incoming data packet and keeps track of incoming traffic rate for each path along with the path head id and number of hops as shown in Figure 3.5 under sink-based-traffic-measurement. The sink keeps the traffic rate on a per path basis in the path table. The sample period is one superframe (as defined in Section 3.4.3) and the sink measures the number of incoming packets in one superframe per path. Then, the sink calculates the weighted moving average of the measured traffic rate per path.

The sink computes the schedule (as shown in Figure 3.5 under sink-based-schedule-computation) by allocating time slots per-path rather than on per-node basis. This is because the sink only has the information about the paths and not about the nodes in the paths. This makes the scheme scalable and not coupled to any tree generated by a particular routing scheme; that is, the schedule computation operates on a simple path abstraction of path-end and hop count and not topological routing information. Therefore, the funneling-MAC is agnostic to the routing scheme or routing tree formations. The sink stores per-path state information in a path-table, which is indexed using the path head id; per-path measurement statistics are also maintained in this table. Each entry contains a path head id, number of

```
1:  # sink-based-traffic-measurement
2:  event (Received a packet) {
3:    for (path=0; path<num_path; path++) {
4:      if (path_head_id[path] = packet -> path_head_id) {
5:        sampled_rate[path] += 1
6:        num_hops[path] = packet -> num_hops
7:  } } }
8:  event (End of Sampling period) {
9:    for (path=0; path<num_path; path++) {
10:      traffic_rate[path] = α*traffic_rate[path]+(1-α)*sampled_rate[path]
11:         if (traffic_rate[path] > max_rate) max_rate = traffic_rate[path]
12: } }
13: # sink-based-schedule-computation
14: for (i=0, j=0; i < max_rate; i++) {
15: for (path=0; path<num_path; path++) {
16:   if (i = 0 or traffic_rate[path] >= i) {
17:      scheduled_slot[j] = num_hops[path]
18:      scheduled_path_head_id[j] = path_head_id[path]
19:      j = j+1
20: } } }
21: for (i=0; i < j-1; i++) {
22: if (scheduled_slot[i+1] > 3) {
23:      scheduled_slot[i]=scheduled_slot[i] - (scheduled_slot[i+1] - 3)
24:      if (scheduled_slot[i] < 1) scheduled_slot[i] = 1
25: }
26: total _slot = total_ slot + scheduled_slot[i]
27: if (total_slot > max_slot) scheduled_slot[i] = 0
28: }
29: # sink-based-dynamic-depth-tuning
30: if (beacon_power < max_power) {
31: if (total_slot < max_slot) beacon_power = beacon_power + step
32: else if (beacon_power > min_power) beacon_power = beacon_power - step
33: }
34: # sensor-based-scheduling
35: for (i=1; i< num_field_in_schedule_packet; i++) {
36: for (j=0; j < num_my_path_head; j++) {
37:     if(schedule_packet_path_head_id(i)=my_path_head_id(j)){
38:        my_slot(slot_num + num_hops_from_path_head) = TRUE
39:     }}
40: slot_num = slot_num + num_hops
41: }
```

Figure 3.5: The funneling-MAC algorithm pseudo-code.

hops, and incoming rate. The incoming rate represents the number of packets each path should carry during one superframe. Note that the sink ages each entry every beacon interval and if the table overflows the sink replaces the oldest entry with a new entry.

Slot Allocation Rule: The sink allocates slots to each path using the information in the path table. For example, assume that the traffic rate of a path is k and the number of hops of the path is h. The sink should allocate every node in the path with slots so the sink allocates slots to the path. If the traffic rate of a path is less than 1, the sink does not

Header	**A ; 3**	**B ; 4**	**C ; 3**

Figure 3.6: Schedule packet structure.

follow the above rule, instead, the sink allocates $1 \times$ h slots to the path. The traffic rate can be less than 1 in the case where periodic traffic with data generation rates of less than 1 packet in one superframe or in the case where event-driven traffic happens occasionally. It is shown in Section 3.2 that the funneling effect exists also in light traffic scenarios so there is a motivation to schedule paths which have a traffic rate less than 1. Since traffic rate of the path is low, the sink should allocate the minimum number of slots to the path. The minimum number of slots that the sink can allocate to a node is 1 slot. Therefore, the sink should allocate every node in the path 1 slot so the sink allocates $1 \times h$ slots to the path. This rule turns out to be good because the testbed evaluation result in Section 3.6.6 shows that the funneling-MAC improves the throughput in light traffic scenario compared to pure CSMA.

Simple Spatial Reuse: To enhance the throughput inside the funnel area, the sink considers spatial reuse. It is very difficult to design an optimal spatial reuse scheme without having the complete physical topology information of the network. However, the sink can compute sub-optimal spatial reuse using only the per-path number of hops state information. The funneling-MAC takes this simple sub-optimal approach and reuses the same slot if two nodes are more than 2 hops away from each other. In this case, f-nodes are unlikely to interfere because one of the nodes may back off due to the fact that in the funneling-MAC carrier sensing is used even for the scheduled access. For example in Figure 3.2, the f-nodes A or B can share the same slot with f-node D because they are 3 hops away. In this case, sink based schedule computation allows f-node B to start transmission three slots after f-node As slot (i.e., at the slot which belongs to f-node D). As a result, the computed schedule is as follows: 3 slots are allocated to the path A-F-E-D, and 4 slots to path B-F-E-D.

Once the sink computes the schedule, it broadcasts a schedule packet for all paths in

its path-table immediately after the next beacon. The sink transmits the schedule packet using the same power level that the sink uses for the beacon so all f-nodes in the intensity region are likely to hear the schedule. Because new schedules are not typically sent each beacon interval the sink sets a schedule expected bit in the beacon header. The payload of the schedule packet contains the path head ids of the scheduled paths and the number of slots allocated to each path, respectively. This resulting per-path schedule is stored in a tuple [path head id (2 bytes), number of slot (1 byte)] in the packet payload. For example in the simple schedule packet shown in Figure 3.6 all f-nodes are informed that there are 3 active paths scheduled in the intensity region and that the 3 paths are allocated, 3, 4, and 3 slots, respectively.

F-nodes receive the schedule packet and figure out which slots are assigned to them as shown as in Figure 3.5 under sensor-based-scheduling. Each f-node keeps a table where it stores the path head node ID of each path going through it and the number of hops to the path head when they forward data packets. Using this table, the f-node can figure out which slots are allocated to itself. For example, the entries of path head id, number of hops kept in the node E are A, 2 and B, 2 so the node E understands that it can transmit two slots after As slot and two slots after Bs slots.

3.4.3 Timing and Framing Issues

Once f-nodes receive a schedule packet, they synchronize their communication to the funneling-MAC framing structure, as illustrated in Figure 3.7. F-nodes transmit their scheduled packets at their allocated slots times in the TDMA frame. To enhance the robustness and flexibility of the funneling-MAC, a CSMA frame (random access period) is reserved between two consecutive TDMA frame (scheduled access period) schedules, and carrier sensing is performed even for scheduled transmissions. The combination of a TDMA and CSMA frame forms what we call a superframe. Several superframes are

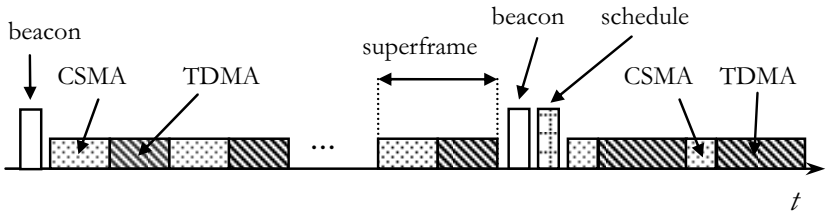


Figure 3.7: Framing.

repeated between two beacons, as illustrated in Figure 3.7, where a schedule packet typically follows the beacon.

The aim of the CSMA frame is to allow for the transmission of the event data packets that have been generated by sensors but have not been allocated slots to be scheduled yet. Other scenarios arise: management, routing, and event data from new nodes suddenly requires transport. One other scenario that is commonly experienced in our testbed is new event data appears on a path due to route changes that occur due to radio vagaries. The sink detects these events using traffic measurements algorithm. Another reason we always offer some CSMA access in the intensity region is to support the transmissions of asynchronous management and control packets such as routing, hello messages, and packet retransmissions for event data packets that are not successfully transmitted during the TDMA frame. Note that the retransmission policy is only an optional part of the funneling-MAC that can be activated if the link reliability should be implemented.

The beacon delivered to f-nodes includes all the necessary frame timing information for the f-nodes to correctly schedule their traffic or contend for the CSMA access in a superframe. Note that from Figure 3.7 the superframe duration is fixed while TDMA duration changes dynamically. The superframe duration has no significant impact on the performance because the sink adapts the schedule to the superframe duration. The sink measures the incoming traffic every superframe and computes the schedule based on the results of sampling process, as described in Section 3.4.2. The TDMA duration changes when the sampled traffic rate at the sink changes. If the traffic load increases sufficiently, the sink allocates more slots

in a superframe so that the TDMA duration grows and more events get scheduled in the intensity region. The portion of a superframe that is not used by TDMA frame is allocated to CSMA frame. In our implementation, we limited the maximum ratio of TDMA/CSMA in a superframe to be 80% so that at least there is a minimum allocation of CSMA to support control packets and unscheduled data packets, as discussed earlier.

The funneling-MAC improves robustness by performing carrier sensing even for scheduled transmissions to avoid possible collisions in transmission anomalies such as in the presence of nodes inside the intensity region that do not receive beacons nor meta-schedule advertisements, as discussed in Section 3.4.4. Finally in terms of framing we note that the funneling-MAC uses the low power listening (LPL) algorithm and preamble technique proposed in B-MAC [80] to reduce energy consumption for sensor networks with low duty cycle. However, unlike B-MAC, f-nodes do not need to transmit a long preamble in LPL mode because their communications are synchronized by the superframe. This frees f-nodes to use the standard short radio preamble. During TDMA access f-nodes wake-up at the beginning of their scheduled listening slot and in the case of CSMA frame f-nodes wake-up periodically based on the wake up periods suggested in [80]. During CSMA access, f-nodes can transmit with the standard preamble because all f-nodes can wake-up and listen at the same time. The nodes outside the intensity region use the long preamble used in LPL mode before transmitting a data.

3.4.4 Meta-Schedule Advertisement

A number of MAC interference issues arise with the funneling-MAC due to its hybrid MAC nature and its broadcasting of sink signaling (i.e., beaconing, schedules) at potentially high power over the complete intensity region. In order not to interfere with any on-going sensor communications in the network (e.g., CSMA forwarding between sensors toward the sink) by such a high power sink transmission, nodes must be capable of learning the superframe

timing details from beacon messages. Another interference issue arises where nodes inside the intensity region may not receive beacons (e.g., due to fading, asymmetric links, etc.) and therefore can become potential interferers by not having the timing and framing information carried in the beacon. One final scenario can occur where nodes outside of the boundary of the intensity region may not be aware of the funneling-MAC frame timing because they do not receive beacons, and as a result, also represent potential interferers. To deal with these interference scenarios (i.e., between scheduled and random access transmissions) the funneling-MAC embeds a low cost meta-schedule advertisement in the first event data packet transmitted by f-nodes, after a new schedule is received.

All f-nodes that received the beacon and schedule embed the meta-schedule in the first event data packet transmitted toward the sink every beacon interval. The mini-schedule contains the following information: superframe duration, TDMA duration, time left of the current TDMA frame, and number of superframe repetitions before the beacon interval expires. The meta-schedule is only 4 bytes in length.

Nodes that are either inside the intensity region and miss a beacon or outside the intensity region but near the boundary can overhear the transmission of meta-schedule carried in a data event. Reception of a meta-schedule allows these nodes to transmit in the CSMA portion of the current superframe mitigating the likelihood of interfering. Now, lets consider a case when an intermediate node of a path inside the intensity region misses a beacon. For example, node F in Figure 2 misses a beacon while the path A-F-E-D is scheduled. The path head f-node A sends a data packet with meta-schedule and node F receives the data packet with meta-schedule. This way, node F can determine that the data packet is scheduled at the current time slot so node F transmits the data packet immediately. Node F uses CSMA frame for its other data packets. Now, let us assume the path A-F-E-D is not yet scheduled and the path head f-node A transmits a data packet with its path information field using CSMA frame. Node F receives the data packet with path information field and node F updates the number of hops field and forwards the data packet so the sink can still

schedule the path A-F-E-D. Therefore, the meta-schedule advertisement allows seamless interoperation between TDMA inside the intensity region, and CSMA operating outside of that region. The use of meta-schedules in this manner resolves potential erroneous behavior.

3.4.5 Dynamic Depth-Tuning

The dynamic depth-tuning algorithm enables the funneling-MAC to maximize the throughput and minimize the packet loss at the sink point. The sink regulates the boundary of the intensity area where TDMA is performed by controlling the transmission power of the broadcast beacon. The sink can dynamically change the transmission power of the beacon and therefore the area in which TDMA is active by determining the optimal depth d of the intensity area in the funnel as shown as in Figure 3.5 under sink-based-dynamic-depth-tuning. The optimal depth analysis in Section 3.6 provides a number of valuable insights that motivate the operation of dynamic depth-tuning algorithm. Section 3.5 shows that the optimal value of d to maximize throughput and minimize packet loss is determined based on the analysis. Based on the analysis in Section 3.5, we propose the following dynamic depth-tuning algorithm. Suppose that A is the total number of slots scheduled, Amax is the number of the maximum available slots in one superframe, and that dmax is the upper bound of the depth d; then the sink chooses d=1 when the network is saturated, that is, where $A > A_{max}$ even with $d = 1$, and if the network is not saturated, then the sink gradually increases d while $A < A_{max}$ and stop increasing d when $A > A_{max}$ or $d > d_{max}$. Since the depth is controlled by the transmission power of beacon signal at the sink, there is an upper bound dmax that matches the maximum transmission power available at the sink. We verified in Section 3.5 that when A=Amax, the depth is at the optimal point where the network achieves both the maximum throughput and minimum loss. This analytical result justifies our approach of adjusting the power to reach that optimality.

The actual operation of dynamic depth tuning algorithm is as follows. When the sink

starts up, it chooses the transmission power as ordinary sensor nodes operating in the network – this is where all the motes and sink use a common power. The sink monitors the channel and computes the schedule with size A as explained in Section 3.4.2. At this point, two different cases may occur: either $A \leq A_{max}$ or $A > A_{max}$. If $A > A_{max}$, then the sink does not increase the transmission power for the next beacon transmission. If $A < A_{max}$, then the sink increments the transmission power of the next beacon by one power level and monitors the performance of channel. The sink keeps incrementing the transmission power in this manner until $A > A_{max}$ or the transmission power reaches its device-limited maximum. If $A > A_{max}$, then the sink decrements the transmission power of the next transmitted beacon by one level. If the transmission power reaches the maximum and $A < A_{max}$, then the sink keeps the transmission power at the maximum. The sink performs this dynamic depth-tuning algorithm on a continued basis, regulating the beacon transmission power accordingly. The pseudo code for dynamic depth tuning algorithm is presented in Figure 3.5 under sink-based-dynamic-depth-tuning.

3.4.6 Low Power Listening

The previous sections describe the funneling-MAC assuming that the low power listening is disabled. The reason is that the main focus of this paper is to address the funneling effect. However, it is necessary to ensure that the funneling-MAC can work together with duty cycling algorithms to improve the energy efficiency without any conflict because the energy efficiency is one of the most important issues in wireless sensor networks. In fact, the funneling-MAC supports a low power listening option. The low power listening should be enabled for sensor networks with low duty cycle to reduce energy consumption of idle listening. The low power listening of the funneling-MAC follows the 'localized' and 'hybrid' approach just like the funneling-MAC access scheme does. The nodes inside the intensity area perform synchronous low power listening while the nodes outside the intensity area

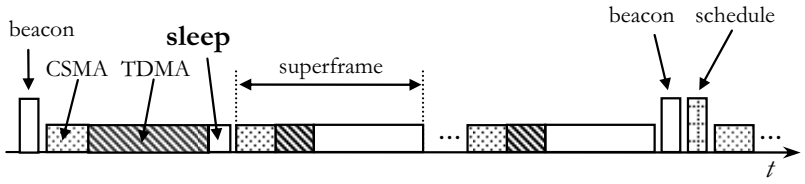


Figure 3.8: Framing for Synchronous Low Power Listening.

perform asynchronous low power listening.

First, the synchronous low power listening of the funneling-MAC is based on the framing illustrated in Figure 3.8. During TDMA frames, each f-node is awake in its scheduled listening slot and sleeps in the time slots that are not allocated to it. During CSMA frames, every f-node is awake and listening. Every f-node sleeps during 'sleep' frames (as shown in Figure 3.8). The size of a sleep frame can be determined based on the traffic load and the delay requirement of the sensing application. To be more precise, the size of a superframe can be determined based on the delay requirement. The size of a TDMA frame and a CSMA frame can be determined based on the traffic load monitored at the sink (as we have discussed it in Section 3.4.3). Then, the remaining time in the superframe is the sleep frame.

Second, the funneling-MAC uses the asynchronous low power listening mode (LPL mode) proposed in B-MAC [80]. However, unlike B-MAC, f-nodes do not always need to transmit a long preamble in LPL mode because their communications are synchronized by the superframe. This frees f-nodes to use the standard short preamble. The nodes outside the intensity region transmit a long preamble in LPL mode before transmitting a packet. The nodes outside the intensity region wake-up periodically based on the wake up periods suggested in [80]. A boundary issue arises with the lower power listening of the funneling-MAC due to its hybrid nature. The outside nodes that are close to the intensity region may not be able to receive the meta-schedule if the f-nodes do not transmit a long preamble. To avoid the problem, the f-nodes transmit a long preamble when they transmit the meta-schedule and the slots in the TDMA frame should be large enough to transmit the

long preamble and the meta-schedule and the data packet. As mentioned in Section 3.4.4, all f-nodes that have received the beacon and schedule embed the meta-schedule in the first event data packet transmitted toward the sink every beacon interval. Therefore, only the slots in the first TDMA frame in each beacon interval need to be large enough to include the long preamble. The size of the slots in the consecutive TDMA frames remains the same as the case in which the low power listening is disabled. Consequently, the sleep frame in the first superframe is smaller than the sleep frame in the consecutive superframes as illustrated in Figure 3.8.

3.5 Optimal Depth Analysis

In this section, we analyze the optimal depth for maximizing the throughput and minimizing the losses so that the dynamic depth tuning algorithm (described in Section 3.4.5) can adjust the depth of the intensity region to this optimal value.

The primary metric we exploit for our analysis is the maximum throughput since many MAC proposals [80] [87] in sensor networks consider throughput as their main performance metric.

The throughput analysis for wireless networks in [41] [62] showed that the overall throughput (i.e., capacity) of a multi-hop wireless network is $O(\sqrt{n})$ if the node density is constant and n is the total number of nodes, which means that the capacity of the network increases as the area size increases. This result is based on an assumption that the source and destination are scattered in the network and spatial reuse is possible. However, in sensor networks the capacity of a network does not increase as the area size of the network increases because all sources have one common destination (which is the sink). The capacity of a sensor network is the maximum number of bits per second that the sink can receive, which is the same as the number of bits per second that the nodes within 1-hop distance from the sink can successfully transmit. These 1-hop nodes are transmitting to one common destination

so no spatial reuse is possible. Therefore, the capacity of a sensor network is $O(1)$, which means that the capacity does not increase as the area size of the network increases.

This observation leads to the conclusion that the throughput of a sensor network is bounded by the capacity of the MAC in the area within 1-hop from the sink. Since the funneling-MAC is a hybrid of CSMA and TDMA, we need to consider the capacity of the two protocols respectively.

The performance analysis of CSMA presented in [9] show that the capacity of CSMA when the channel is saturated (i.e., every node has always something to send) is in the form:

$$S_c = f(n, W_{min}, m), \tag{3.1}$$

where n is the number of contending nodes, W_{min} is the minimum contention window size, and m is maximum backoff stage. The maximum utilization of TDMA when the channel is saturated is in the form:

$$S_t = f(t_s, E_p), \tag{3.2}$$

where t_s is the size of a time slot and E_p is the average packet size.

In the funneling-MAC, W_{min}, m, t_s, and E_p are constant. The capacity of the funneling-MAC in a given network is:

$$C_f = r \cdot S \cdot \frac{A_1}{A} + (1 - r) \cdot S_c \cdot \frac{B_1}{B_1 + B_2}, \tag{3.3}$$

where the constant r is the ratio of TDMA frame in the superframe, A_1 is the number of TDMA slots allocated to the nodes that are 1 hop away from the sink, A is the total number of allocated TDMA slots, B_1 is the amount of CSMA transmission opportunities that are given to the nodes that are 1 hop away from the sink, B_2 is the amount of CSMA transmission opportunities that are given to the nodes that are 2 hop away from the sink. The first component $r \cdot S_t \cdot \frac{A_1}{A}$ is the capacity of TDMA frame and $\frac{A_1}{A}$ indicates the ratio of

the capacity that is given to the first hop nodes over the total TDMA frame capacity. While the component of equation $(1 - r) \cdot S_c \cdot \frac{B_1}{B_1+B_2}$ is the capacity of CSMA frame and $\frac{B_1}{B_1+B_2}$ indicates the ratio of the capacity that is given to the first hop nodes over the total CSMA frame capacity. From the slot allocation rule in Section 3.4.2, A and A_1 can be calculated as:

$$A = w \cdot \sum_{i=1}^{d} i \cdot N_i, \tag{3.4}$$

$$A_1 = w \cdot \sum_{i=1}^{d} N_i, \tag{3.5}$$

where d is the depth (in number of hops) and N_i is the number of nodes that are i hop away from the sink and w is the weight function based on the monitored traffic. The weight function w is same for all paths since every node always has something to send when the network is saturated (as defined in [9]). In equation 3.3, only $\frac{A_1}{A}$ is the function of d (as indicated by equation 3.4 and 3.5) and other variables are not related to d. Thus from equation 3.3, the value of d that maximizes the throughput C_f is the value of d that maximizes $\frac{A_1}{A}$. For all integer values of i from 1 to d, the value of $i \cdot N_i$ is always greater than the value of N_i, so the value of $\frac{A_1}{A}$ decreases as the value of d increases. Hence from equations 3.4 and 3.5, we can find that $\frac{A_1}{A}$ is maximized when $d = 1$. In conclusion, the capacity of the funneling-MAC C_f is maximized when the depth $d = 1$. The more we increase the depth from 1, the larger is the capacity drop. This might be a surprising result but if we consider the funneling effect, the area within 1 hop from the sink is the bottle neck of the network and is where contention-free access is needed the most.

Now, lets consider the case when the network is not saturated such that the capacity is not an important issue anymore. In this case, we are interested in reducing the number of collisions in the network. It is natural to think that by maximizing the use of TDMA the number of collisions can be minimized. Hence we choose to increase the depth when channel is not saturated. However, in the funneling-MAC, the number of slots required in

one superframe increases as the depth increases, as shown in equation 3.4, because of the increase on the number of hops for each path. Consequently, the number of slots required would exceed the available number of slots at some point, in which case, packets can be lost during TDMA frame. Note that the loss rate for the funneling-MAC during CSMA frame is independent from the value of the depth d because all the transmissions during this period use CSMA regardless of the depth of the intensity area.

The number of slots required in one superframe with depth d is presented in equation 3.4 and the number of available slots in one superframe time t_f is:

$$A_{max} = r \cdot \frac{t_f}{t_s}, \tag{3.6}$$

where the constant r is the ratio of TDMA frame in the superframe and t_s is the size of a time slot as we mentioned earlier. The loss rate for the funneling-MAC with depth d during TDMA frame is:

$$L_f = P_s \cdot \frac{S}{S + S'} + P_c \cdot \frac{S'}{S + S'} + P_e, \tag{3.7}$$

where P_s is the probability of loss due to schedule overflow, P_c is the probability of loss due to collision outside the intensity region, and P_e is the probability of loss from other occasions (e.g., radio error, system error, etc), S is the number of scheduled transmission inside the intensity area, and S is the number of contention-based transmission outside the intensity area. To simplify the analysis, we assume that P_e is constant and P_c is a function of the depth d (and not impacted by any other factors). We observed in Figure 3.4 that the probability of loss for CSMA decreases as the distance from the sink increases. To analyze the impact of the funneling effect observed in Figure 3.4, we assume that P_c decreases as the distance from the sink increases. Hence, P_c decreases as the depth d increases. When $A \leq A_{max}$, then $P_s = 0$ so the loss L_f decreases as the depth d increases. At some point, the increase of the depth d will cause $A > A_{max}$, in which case, the funneling-MAC just drops

the over-allocated $A - A_{max}$ packets so $P_s = \frac{A-A_{max}}{A}$.

Lets consider the two cases when $A = A_{max} + \Delta$ and when $A = A_{max}$. Note that the variable Δ, which is a positive integer, is introduced to understand the response to the additive increase of the depth d. The difference between the loss rates of the two cases (when $A = A_{max} + \Delta$ and when $A = A_{max}$) is:

$$L_\Delta = L_{A_{max}+\Delta} - L_{A_{max}}$$

$$= \frac{\Delta}{A_{max}+\Delta} \cdot \frac{A_{max}+\Delta}{T} + P_{c1} \cdot \frac{T - A_{max} - \Delta}{T} - P_{c2} \cdot \frac{T - A_{max}}{T}$$

$$= \frac{\Delta}{T} \cdot (1 - P_{c1}) + \frac{T - A_{max}}{T}(P_{c1} - P_{c2}), \tag{3.8}$$

where P_{c1} is the probability of loss due to collision when $A = A_{max} + \Delta$, and P_{c2} is the probability of loss due to collision when $A = A_{max}$, and T is the total number of transmission including the case when packets are dropped due to over-allocation. As mentioned above, the probability of loss P_c decreases as the depth d increases, thus $P_{c1} \geq P_{c2}$. Because $P_{c1} \geq P_{c2}$ and $P_{c1} \leq 1$ (as P_{c1} is a probability variable that can only have values from 0 to 1), $L_\Delta \geq 0$ for any $\Delta > 0$. Therefore, the value of L_f when $A > A_{max}$ is always greater than the value of L_f when $A = A_{max}$.

In conclusion, the loss rate can be minimized when $A = A_{max}$ and at this optimal point the loss is guaranteed to be smaller than pure CSMA system. The difference in terms of loss rate between the funneling-MAC and pure CSMA system is $P_{c2} \cdot \frac{A_{max}}{T}$ at this optimal point.

3.6 Sensor Testbed Evaluation

We take an experimental approach to the evaluation of the funneling-MAC and present a number of experiments that give insights into the performance tradeoffs of the protocol under a wide variety of systems conditions, e.g., different traffic conditions, different mote

Table 3.1: Funneling-MAC experimental parameters.

Parameter	Value
Default data transmission power (Cdata)	-10 dBm
Beacon and schedule transmission power (Ccontrol)	-10 5 dBm
Step size of power for dynamic depth-tuning (Cunit)	1 dBm
Beacon interval (tb)	20 sec
Superframe size (tf)	1 sec
Slot size (ts)	30 msec
Moving average factor ()	0.9

topologies and densities (from simple benchmarks to more realistic dense grid), and compare the performance of the funneling-MAC to the baseline TinyOS B-MAC protocol [100] and the Z-MAC implementation on TinyOS [99].

3.6.1 Experiemental Set-up

We implement the funneling-MAC on mica-2 motes using the default TinyOS [100] MintRoute routing protocol and Surge applications to drive different source rates. The bit rate of the radio interface for mica-2 motes is 19.2 kbps. Our experimental testbed comprises of a 45 mote dense grid deployed in a large laboratory room and is configured, as shown in Figure 2 unless specified otherwise. Node spacing and transmission power of the sensors are set such that one-hop neighbors achieve >80% delivery, while two-hop neighbors achieve <20% delivery. In this way, a fairly strict and dense multi-hop radio environment is constructed for experimentation. We use the default TinyOS packet size, which is 36 bytes.

We implement the funneling-MAC on top of B-MAC, which provides the baseline CSMA system. Note, that we do not use fixed routes as in [87] because we are interested in how well the protocols under comparison, B-MAC, Z-MAC, and the funneling-MAC perform in a realistic networking scenario where time-varying radio conditions can impact coverage, link quality, and routing paths. For B-MAC and Z-MAC, we use the default

settings described in [80] [87], respectively. The parameter settings of the funneling-MAC are presented in Table 3.1. The settings that are not specified in Table 3.1 are the settings used in [80] as the funneling-MAC is built on top of B-MAC. For all experiments, we turned off the low power listening and use the same preamble size for B-MAC, Z-MAC, and the funneling-MAC for fair comparison. We adjusted the data transmission power of sensor nodes at -10 dBm in order to build up a strict multi-hop network (up to 5 hops), as discussed in Section 3.2. The funneling-MAC dynamically tunes the power of beacon and schedule at the sink node from -10 dBm to 5 dBm (i.e., the maximum transmission power of the CC1000 transceiver [23]) in increments or decrements the power of 1 dBm which is the unit power level, as reported in [23].

The beacon interval is initially computed based on the motes clock accuracy and the required accuracy of synchronization for scheduling on the media. We run some experiments with various values for the beacon interval and we experimentally determine a beacon interval of 20 seconds gives the best performance in terms of throughput with the necessary accuracy. We also experiment with lazy-beaconing where we trade performance for a larger beacon interval. We observed that we can push the beacon interval out to 50 seconds with only a marginal drop in throughput performance. However, for beacon intervals greater than 50 seconds we register a sharp reduction in throughput measured at the sink of approximately 30%, showing that the loss of scheduling accuracy and schedule drift is too costly for the further reduction in signaling overhead. For the experiments reported in this section we chose a beacon interval of 20 seconds for increased scheduling accuracy and to remove any likelihood of schedule drift. Table 3.1 shows the set of experimental parameters for the funneling-MAC testbed that are consistently applied across all experiments.

3.6.2 Impact of Depth-Tuning

We are interested in evaluating the impact of the depth of the intensity region on the measured throughput of the sensor network testbed for the following reasons. First, in order to verify that by pushing the TDMA area (i.e., the intensity region) beyond the optimal depth will only degrade in measured throughput at the sink. Second, to show that the dynamic depth-tuning algorithm is valid when implemented in a real sensor testbed. To compare dynamic depth-tuning to the simple case of just scheduling the last hop (i.e., one hop from the sink) we fix the dynamic depth-tuning algorithm to one hop only. Note, that the results in Section 3.2 indicate that most packet loss occurs over the last hop to the sink. Following this logic, we consider a baseline algorithm as having a fixed depth of one, which only schedules the last hop, and an optimized algorithm that schedules additional hops using the fully enabled dynamic depth tuning algorithm. In what follows, we show that the optimized algorithm achieves considerably better performance than the simple baseline algorithm does.

In order to observe the impact of depth on performance, we fix the beacon transmission power to the values of -10, -8, -6, -4, 0, and 4 dBm, respectively. The depth of the intensity region is an approximate function of the beacon transmission power used. In essence, we can approximate the depth in terms of the beacon transmission power coverage distance in terms of number of hops from the sink for our grid network. For example, if the sink transmits a beacon using the default transmission power of ordinary sensor nodes, this will approximate coverage of one hop from the sink. Likewise, we can expect that a beacon will have a greater coverage than one hop for higher transmission powers. The metric that we observe with each beacon power setup is the throughput. We define the choke point throughput of the sink as the amount of data in terms of bits received at sink over a 1 second period. In these experiments, all 44 nodes are sources. We run experiments for 3 different source rates low, medium, high: 0.2 pps, 1 pps, 2 pps, respectively.

We plot the results in Figure 3.9. For each of the source rates we measured the sink

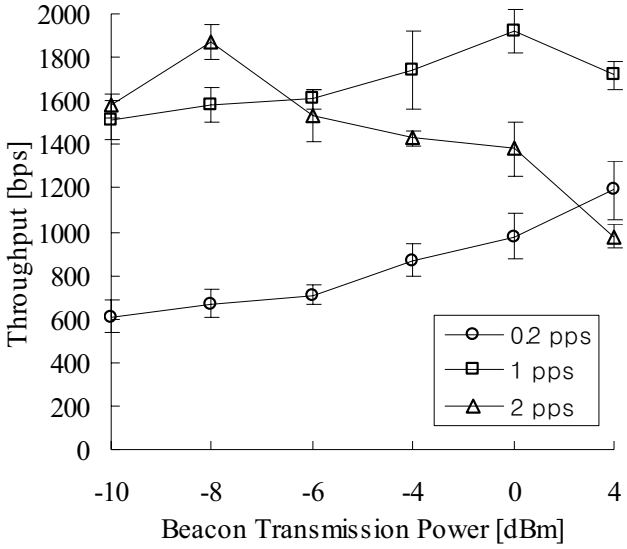


Figure 3.9: The impact on throughput over varying depth in terms of beacon transmission power.

throughput for increasing beacon power (which approximates the depth of the intensity region coverage). The result indicates that there is an approximation of the optimal transmission power for beacons (i.e., optimal depth) that maximize the throughput such that if we use a larger transmission power than the optimal power, the throughput measured at the sink degrades. This means that if we increase the TDMA area further the optimal depth by using more power then it degrades the measured throughput.

Figure 3.9 validates the dynamic depth-tuning algorithm. According to the analytical result in the Section 3.5, the optimal depth is close to 1 hop (i.e., the beacon transmission power is the same as the motes data event transmission power) when the network is saturated, while the optimal depth is greater than 1 hop when the network is not saturated. In fact, if we set the source rate to 2 pps, which drives the network toward saturation, the optimal beacon transmission power from our experimental result is -8 dBm, which provides radio coverage close to 1 hop (i.e., the motes data event transmission power of -10 dBm). We observe in Figure 8 that the optimal depth is greater than 1 hop when the network is not saturated (i.e., 0 dBm for 1 pps, and 4 dBm or greater for 0.2 pps). These experimental

observations validate the analytical observation in the Section 3.5 and thus provide a sound basis for the dynamic depth-tuning algorithm.

In what follows, we quantify how much gain the baseline and optimized algorithms can achieve over B-MAC. From Figure 3.19(c), we can observe that the throughput of B-MAC for 0.2 pps, 1 pps, and 2 pps source traffic rates is 272 bps, 1099 bps, and 1631 bps, respectively (we discuss this plot further in Section 3.6.5). The throughput related to the -10 dBm x-axis value in Figure 3.9 (i.e., 1583 bps for 2 pps, 1511 bps for 1 pps, and 645 bps for 0.2 pps) represent the performance of the funneling-MACs baseline algorithm that schedules only the last hop with the depth fixed by -10 dBm beacon power. The throughput shown in Figure 3.9 at the optimal beacon transmission powers (i.e., 1872 bps at -8 dBm for 2 pps, 1925 bps at 0 dBm for 1 pps, and 1191 bps at 4 dBm or greater for 0.2 pps) represent the performance of the funneling-MACs optimized algorithm (i.e., when dynamic depth-tuning is fully enabled). The gain over B-MAC for the baseline algorithm with 0.2 pps, 1 pps, and 2 pps is 124%, 37%, and 0%, respectively. The gain over B-MAC for the optimized algorithm with 0.2 pps, 1 pps, and 2 pps is 338%, 75%, and 15%, respectively. For all source traffic rates (viz. 0.2 pps, 1 pps, and 2 pps) the optimized algorithm performs better than the baseline algorithm. More importantly, the baseline algorithm does not achieve any gain over B-MAC when the source rate is 2 pps. This result indicates that the baseline algorithm provides some gain that may be sufficient for simple low complexity deployments (i.e., schedule only the last hop) but the optimized algorithm provides considerably better performance despite that the optimized algorithm comes with some added complexity over the baseline algorithm. As a result, we recommend using dynamic depth-tuning in its fully enabled form as a default.

3.6.3 Impact of Boundary Node Interference

In what follows, we show that the meta-schedule advertisement is effective at dealing with the interference scenarios discussed in Section 3.4.4. We study the impact of abruptly changing the depth of the intensity region on boundary node behavior and the measured sink throughput performance. In this experiment meta-schedule advertisements exploit the broadcast nature of the radio medium, where nodes receive the embedded meta-schedule simply by overhearing data event packets with embedded meta-schedules sent by neighboring nodes. The use of meta-schedule allow for the co-existence of TDMA inside the intensity region and pure CSMA outside that region.

We evaluate the behavior of nodes at the boundary of the intensity region for some interfering scenarios. We set up an experiment that studies the impact of boundary variability. In this experiment the sink changes the beacon transmission power for every beacon by selecting the transmission power between two values in turn. We choose the two beacon transmission power values -6 dBm and -8 dBm such that the boundary of the intensity region falls approximately across the center of the grid testbed where there is a higher density of nodes that will be included in TDMA scheduling (at -8 dBm) and then dropped out (at -6 dBm) as they fall outside of the intensity region and operate without the framing and timing information.

We run the experiment of switching between -6 dBm and -8 dBm for a number of different source data rates. Figure 3.10 shows the various source rates and the corresponding throughput performance measured at the sink. This is for the case where all the 44 motes are sources. We study two experiments, one called variable power where the transmission is alternating between -6 dBm and -8dBm, and one called fixed power where we fix the beacon transmission power to -7 dBm which represents the average of the variable case. The comparison of the throughput measured on each experiment is shown in Figure 3.10. We run the experiment five times for each data rate and calculate 95% confidence interval. From

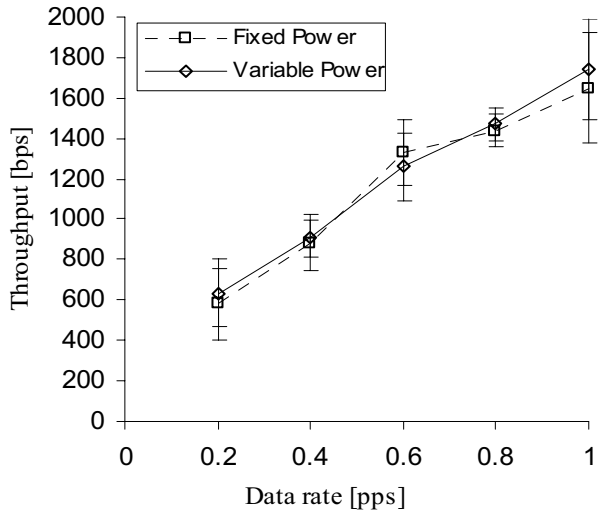


Figure 3.10: Throughput with fixed/variable beacon transmission power.

the plot we can see that the measured throughput for fixed and variable power cases are almost the same (i.e., within the confidence interval of each other). This result indicates that boundary variability stressed in this test has little impact on the ability of the funneling-MAC to operate stably. As part of this test we instrument the motes to record if the beacon timeout occurred and the mote had no framing information but overheard meta-schedules. We found that 8% of the boundary motes fall into this category; that is, motes that are consistently inside and outside of the intensity region as the beacon transmission power toggled between -6 dBm and -8 dBm at the beacon interval. This indicates that these 8% of nodes would have become interfering modes if they had not successfully overhead embedded meta-schedule advertisements.

3.6.4 One-hop and Two-hop Benchmark

Our experiment setup in this section reproduces the one-hop and two-hop throughput benchmark described in [80] and [87]. We run the benchmarks to verify the correctness of our testbed setup by comparing the results achieved to the B-MAC and Z-MAC benchmark performance. For the benchmarks and the experimental evaluation of the funneling-MAC,

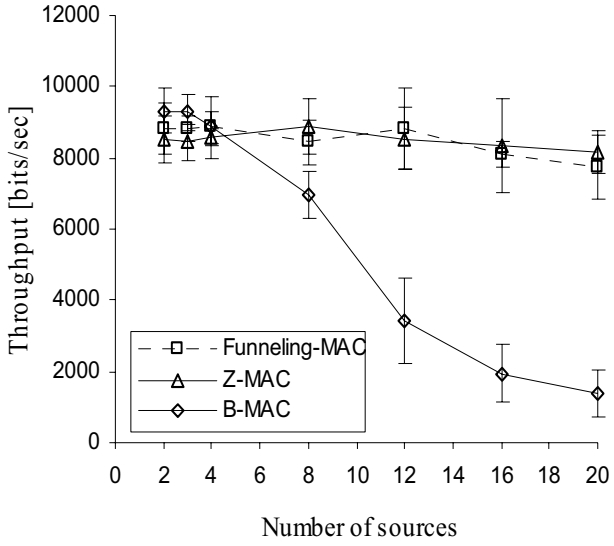


Figure 3.11: Throughput comparison in the one-hop benchmark.

Z-MAC, and B-MAC we use the MintRoute routing. Note, that all the funneling-MACs algorithms including the dynamic depth-tuning mechanism are enabled throughout the experimental evaluation.

The goal of the one-hop benchmark is to measure the maximum throughput achievable by the funneling-MAC, Z-MAC, and B-MAC. In either the benchmarks the throughput is measured as a function of the number of contenders. In the one hop benchmark a set of nodes is located such that the nodes are all within the same contention area. The nodes are placed in a circle where the sink is in the middle. Each node transmits as fast as possible without duty cycling and we measure the throughput at the sink. The result is shown in Figure 3.11. From Figure 3.11 we derive the correctness of our set-up since the throughput achieved with Z-MAC and B-MAC present the same pattern reported in [80] and [87], respectively. However, the average achievable throughput is a little less than the throughput presented in [80] and [87] and we believe this is mainly due to difference of the environments where the testbeds are located. Given their hybrid TDMA/CSMA nature both the funneling-MAC and Z-MAC noticeably reduce the amount of collisions as shown in Figure 3.11 resulting

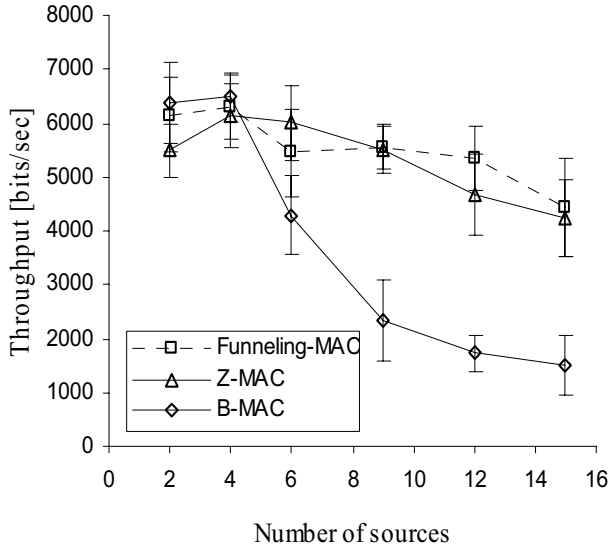


Figure 3.12: Throughput comparison in the two-hop benchmark.

in a considerable better performance over B-MAC as the number of contenders increases. The probability of collision is small with few contenders and this is the reason why B-MAC, Z-MAC, and the funneling-MAC present almost the same performance when the number of sources is small (less than 5).

The two-hop benchmark reproduces the setup used in [87] and its goal is to evaluate the robustness of the protocols against hidden terminals. Nodes are placed in two different clusters and the sink is placed between the clusters. We adjust the transmission power of the nodes in order to make nodes located in different clusters hidden to each other. Nodes transmit as fast as they can and we measure the throughput at the sink placed between the two clusters. In Figure 3.12 it is shown that the throughput achieved by Z-MAC and B-MAC in our two hop benchmark set-up present the same pattern as in [87]. B-MAC suffers packets collisions due to medium contention and hidden terminal problem so its throughput drops off as the number of sources increases.

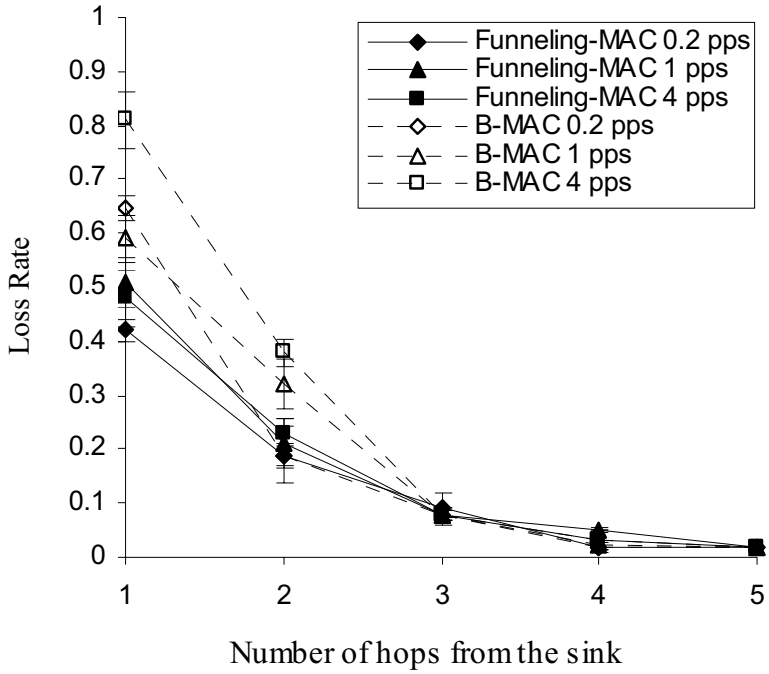


Figure 3.13: Loss rate over varying distance from the sink for B-MAC and the funneling-MAC.

3.6.5 Loss Rate Distribution

In Section 3.2, we quantify the impact of the funneling effect on the packet loss rate distribution for B-MAC. In what follows, we now assess the impact of the funneling effect on the funneling-MAC. We use the same setup (i.e., multi-hop testbed using 45 motes) and metric (i.e., loss rate) as in Figure 3.4. The result is presented in Figure 3.13. For the comparison, we also include the B-MAC result in the figure. Figure 3.13 shows the loss rate across an increasing number of hops from the sink. We observe that the funneling effect is mitigated by comparing the steepness of the slopes of the funneling-MAC and B-MAC curves, respectively. The loss rate over the first two hops from the sink is significantly different because the funneling effect is active, before both curves converge on the same performance at three hops from the sink where the funneling effect is no longer present in the experiments. After this point both MACs offer similar CSMA performance for 3, 4,

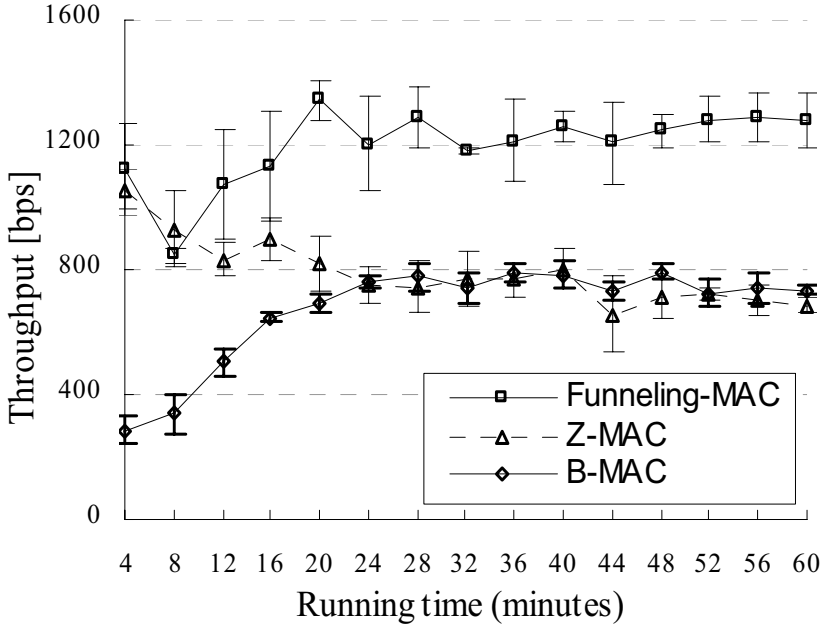


Figure 3.14: Trace of throughput over running time for the funneling-MAC, B-MAC, and Z-MAC (Dartmouth College Testbed).

and 5 hops from the sink. However, the loss rate over the first two hops is considerably smaller when using funneling-MAC over B-MAC. For example, at the higher source rate of 4 pps B-MAC's loss rate is 81% at one hop and 40% at two hops from the sink, while the funneling-MAC reduces those loss rates to 48% at one hop and 22% at two hops for the same source rate. The loss rate for the funneling-MAC remains almost the same for varying source traffic rates (viz. 0.2 pps, 1 pps, 4 pps) while the loss rate for B-MAC varies considerably with source rate and across the first two hops from the sink. In the following section, we show how this reduction of loss rate in the first few hops impacts the overall throughput performance of the sensor network.

3.6.6 Multi-hop Throughput

In this experiment, we compare the throughput of B-MAC, Z-MAC, and the funneling-MAC in our multi-hop testbed consisted of 45 motes as explained in the section 3.6.1.

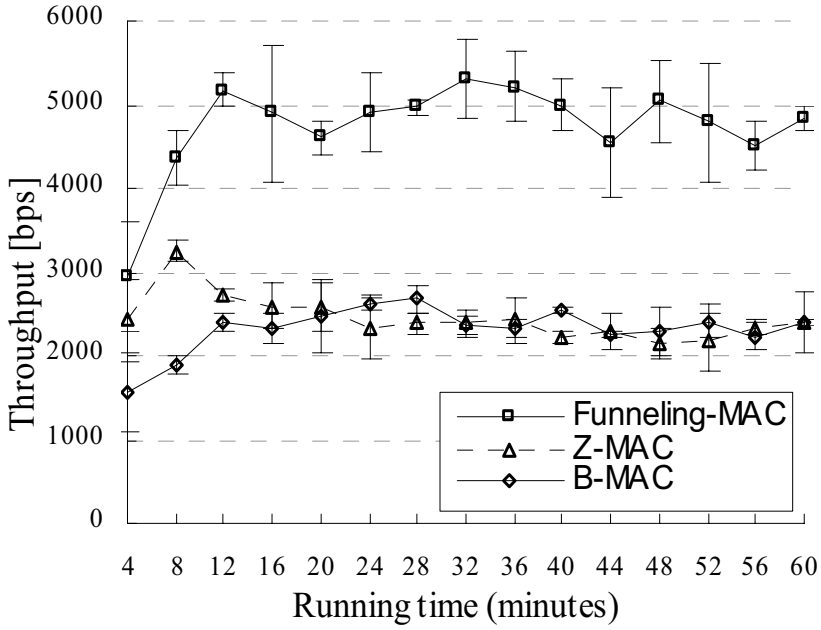


Figure 3.15: Trace of throughput over running time for the funneling-MAC, B-MAC, and Z-MAC (Columbia University Testbed).

Figure 3.14 shows the trace of the throughput of the funneling-MAC, Z-MAC, and B-MAC over time for the experiment where all 44 nodes are sources generating 5 pps. This scenario represents a heavy traffic load. We run the experiments 5 times with this setup and compute the average throughput with 95% confidence interval. At start of the experiment, Z-MAC and the funneling-MAC perform equally while B-MAC performs worst. It is worth noting that the routing paths from all sources are not completely established until approximately 20 minutes into experiment. Protocols such as MintRoute take a significant amount of time with the default settings to create sufficient routing state [102] before the performance of the network stabilizes at around 20 minutes into the experiment. When a node does not have a route it sends the event data to the broadcast channel, which contributes to congestion and degrades the throughput further. As more source nodes acquire routes and path, the funneling-MAC and B-MAC gain performance in terms of throughput. The funneling-MAC outperforms B-MAC consistently over time.

Schedule Drift: We can observe from Figure 3.14 that Z-MAC throughput steadily

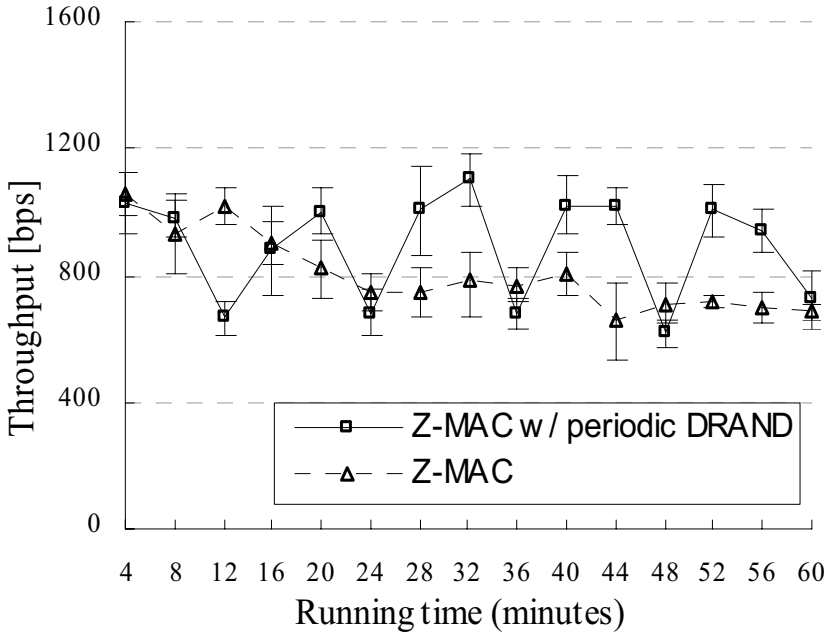


Figure 3.16: Trace of throughput over running time for Z-MAC with/without periodic DRAND (Dartmouth College Testbed).

degrades as time increases. In [87] it is noted that Z-MAC runs DRAND only at the beginning and not periodically. Hence it is possible that the reason Z-MAC degrades is because it is susceptible to schedule drift where the initial schedule computed by DRAND is no longer valid due to time-varying radio conditions and possibly route changes, forcing Z-MAC to fall back to the performance of CSMA, as shown in the plot. To verify that the Z-MAC throughput degradation is not a product of our dense grid sensor testbed setup at Dartmouth College we ran the same experiment on a more sparse larger area sensor testbed at Columbia University, as shown in Figure 3.18. The Columbia University testbed consists of 31 mica-2 motes and their placement is shown in Figure 3.18. The transmission power of each mote is set to -10dBm and, at this power, on average all nodes have at least 7 nodes from which the packet delivery ratio is at least 80%. The results from the Columbia University testbed are presented in Figures 3.15 and 3.17. Figure 3.15 shows that the Z-MAC throughput degradation is reproducible on the Columbia University testbed although the scale and timing of the degradation are different.

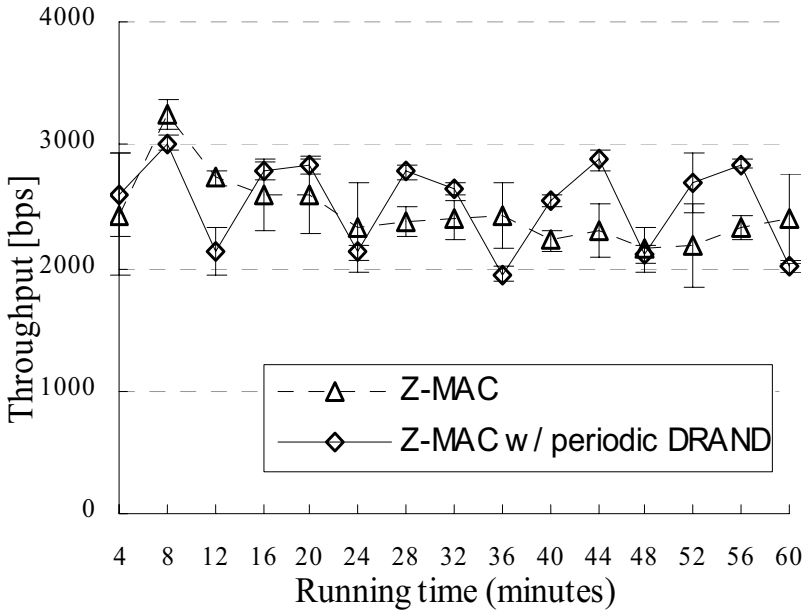


Figure 3.17: Trace of throughput over running time for Z-MAC with/without periodic DRAND (Columbia University Testbed).

To confirm that the throughput degradation is due to schedule drift, we run Z-MAC with DRAND running at the start of the experiment and then every 12 minutes. After each DRAND run, each node reports its neighborhood table to the sink. We analyzed these reports of neighbor table from each node. Each node keeps a table that contains the IDs of its neighbors in its two-hop transmission range. The schedule (computed by DRAND) is the result of the interaction between the nodes within the two-hop transmission range. Hence, the neighbor table is a good indication of the validity of the schedule. Analyzing these reports, we observe that 76.7% of the nodes experience some changes in their neighbor table after each DRAND re-run in the Dartmouth College testbed and 59.6% in the Columbia University testbed.

In order to quantify the degree of changes in neighborhood, we define a metric as

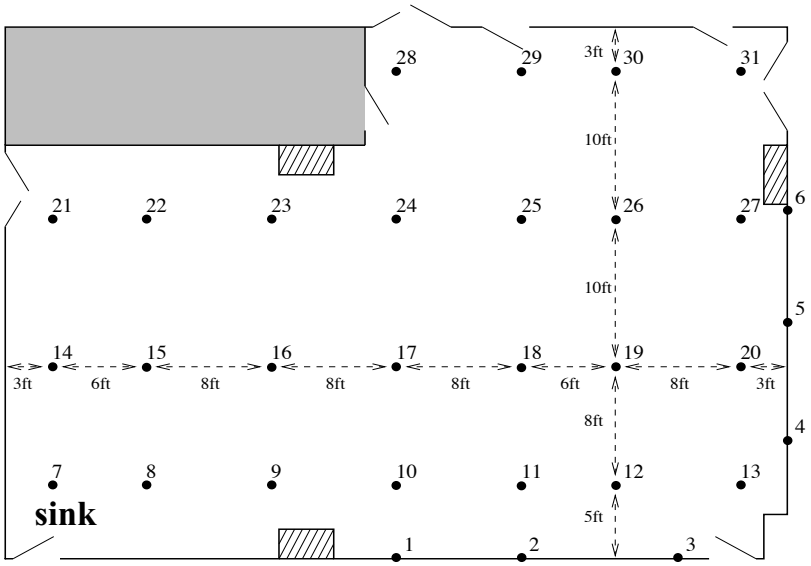


Figure 3.18: Columbia University testbed where 31 Mica2 motes are mounted at the labeled positions across the ceiling of a 1600 ft^2 room.

following. The degree of changes in the neighbor table of a node is as follows:

$$X_{ij} = \frac{D_{ij} + D_{ji}}{H_{ij}} \tag{3.9}$$

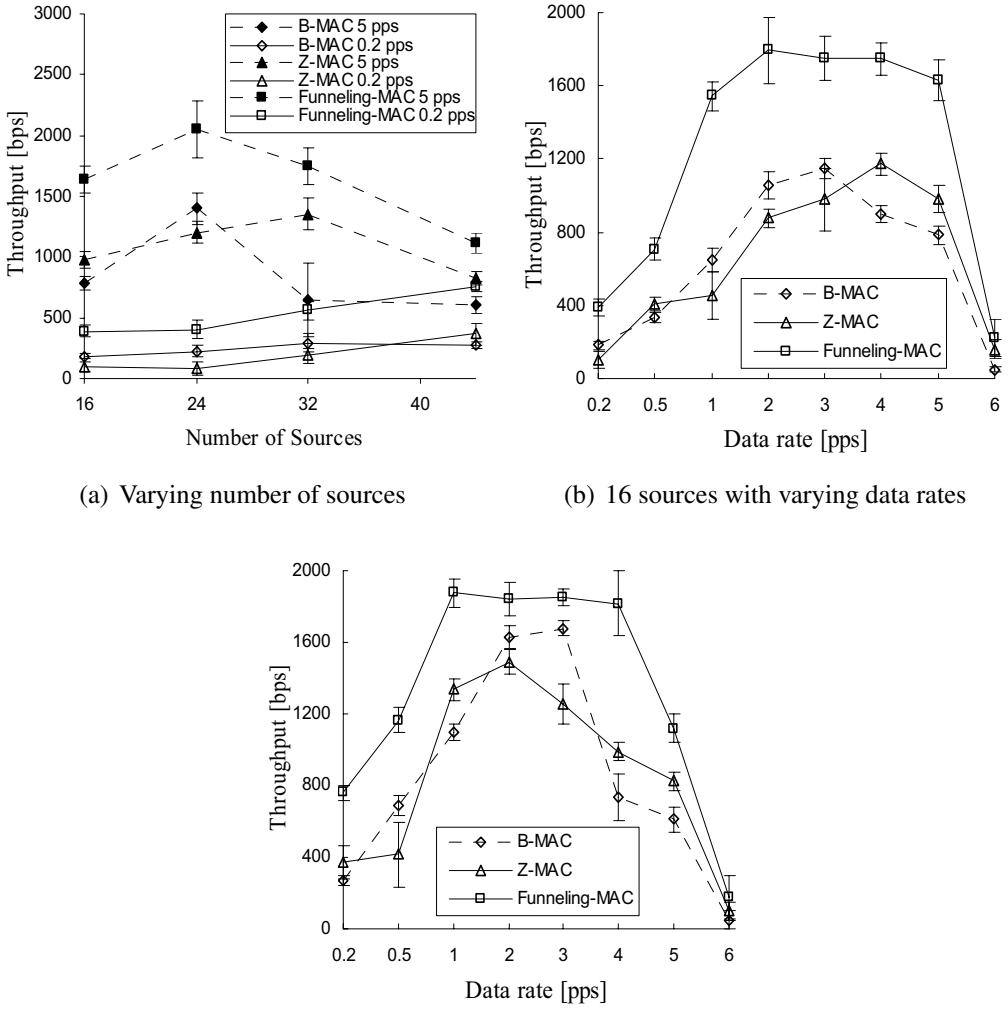
where T_i is the neighbor table of the node after the i-th DRAND run, D_{ij} is the number of nodes that are in neighbor table T_i but not in table T_j, and D_{ji} is the number of nodes that are in neighbor table T_j but not in T_i, and H_{ij} is the number of nodes that are in neighbor table T_i OR T_j.

After calculating X_{12}, X_{23}, X_{34}, and X_{45} of each node, we calculated the average of X_{12}, X_{23}, X_{34}, and X_{45} among the nodes. From the experiment in Dartmouth College Testbed, the average of X_{12} (i.e., the degree of changes between the first DRAND run at initial time and the second DRAND run at 12 minute) is 45.5%. The average of X_{23} is 25.3%, the average of X_{34} is 30.5%, and the average of X_{45} is 31.0%. In the Columbia University Testbed, the average of X_{12}, X_{23}, X_{34}, and X_{45} is 53.7%, 25.9%, 45.7%, and 30.5%, respectively.

The fact that the majority (76.7% and 59.6%) of the nodes have experienced some

changes in their neighbor table and the fact that average degree of changes are considerable (ranging from 25.3% to 53.7%) indicates that the schedule computed by DRAND at the start of the experiment does not take into account the changes in neighborhood over the time. The neighborhood change is due to radio characteristic variability and environmental factors. In particular, as shown in [52] as the traffic in the network increases the link quality consistently decreases. This implies that the link quality used during the DRAND phase to determine the two hop neighborhood membership will not be the same as the link quality during the data transmission phase since the link quality decreases as the traffic increases. Hence the two hop neighborhood perception during the DRAND phase is likely to be different from the two hop neighborhood perception during the data transmission phase.

The throughput comparison between Z-MAC without periodic DRAND and Z-MAC with periodic DRAND is shown in Figure 3.16 (Dartmouth College testbed) and Figure 3.17 (Columbia University testbed). The solid lines in Figures 3.16 and 3.17 indicate that Z-MAC with periodic DRAND does not suffer from the throughput degradation except for the time DRAND is running. During each DRAND run (at 12, 24, 36, and 48 minute in the experiment), the throughput degrades below 700 bps in Figure 3.16 and below 2200 bps in Figure 3.17. At the beginning of DRAND run, the DRAND neighbor table is initialized (zero entry in the table), so the network operates just like B-MAC during DRAND run. In addition, the signaling overhead of DRAND contributes to the degradation. It is shown in the Section 3.6.7 that the signaling overhead of DRAND is large. As soon as the DRAND run is complete, the throughput returns back to 1000 bps or greater in Figure 3.16 and 2600 bps or greater in Figure 3.17; the value of which is the same level as the throughput just before DRAND run begins. This result (that Z-MAC with periodic DRAND does not suffer from throughput degradation) indicates that the schedule computed by DRAND running every 12 minutes is responsive to neighbor changes. Hence, this result indicates that the cause of throughput degradation is due to the schedule drift where the initial schedule computed by DRAND is no longer valid.



(a) Varying number of sources

(b) 16 sources with varying data rates

(c) 44 sources with varying data rates

Figure 3.19: Throughput comparison of the funneling-MAC, Z-MAC, and B-MAC.

Figure 3.19 compares the throughput of the funneling-MAC, Z-MAC, and B-MAC over varying number of sources and data rates. We varied the number of sources each time we run an experiment. When the number of sources is less than 44, the sources are randomly chosen among 44 sensor nodes in the grid. We also varied the packet generation rates of the sources each time we run a test. The x-axis values represent the number of packets generated per each source node.

From Figure 3.19 we note that the overall throughput results show that the funneling-

MAC consistently achieves greater throughput than B-MAC and Z-MAC under various conditions, such as changing number of sources and changing data rates (from very low rates to very high rates). The throughput curves for Z-MAC and B-MAC shown in Figure 3.19 follow the same general pattern in [87]. For light traffic, Z-MAC and B-MAC perform the same, while for high loads Z-MAC outperforms B-MAC. Note that for data rates higher than 5 pps, MintRoute protocol cannot setup routing paths for any node, thus, after this rate the network goes into collapse as clearly indicated in Figure 3.19(b).

There are two reasons why Z-MAC only shows marginal improvements over B-MAC in the presence of the funnel effect while the funneling-MAC outperforms both B-MAC and Z-MAC by large margin. One of the reasons is the schedule-drift associated with Z-MAC. The other reason is currently that DRAND cannot take into account the funneling effect and the need to allocate more slots to nodes in the intensity region. This is because DRAND is a pure distributed coloring algorithm that does not take into account the existence of the central entity (i.e., the sink) and the funneling traffic pattern (i.e., the funneling traffic pattern that every data packets are flowing toward the sink). Hence, DRAND can only allocate the same amount of slots among its two hop neighbors. In contrast, the funneling-MAC is capable of allocating slots based on the funneling traffic pattern so that the funneling-MAC allocates more slots to the nodes closer to the sink providing a big win in performance of the measured throughput at the sink.

We also note that the funneling-MAC performs better than Z-MAC and B-MAC even under light traffic conditions where the funneling effect is very evident. So even under light load B-MAC and Z-MAC are not capable of mitigating the negative effects of funneling.

3.6.7 Energy Tax and Signaling Overhead Cost

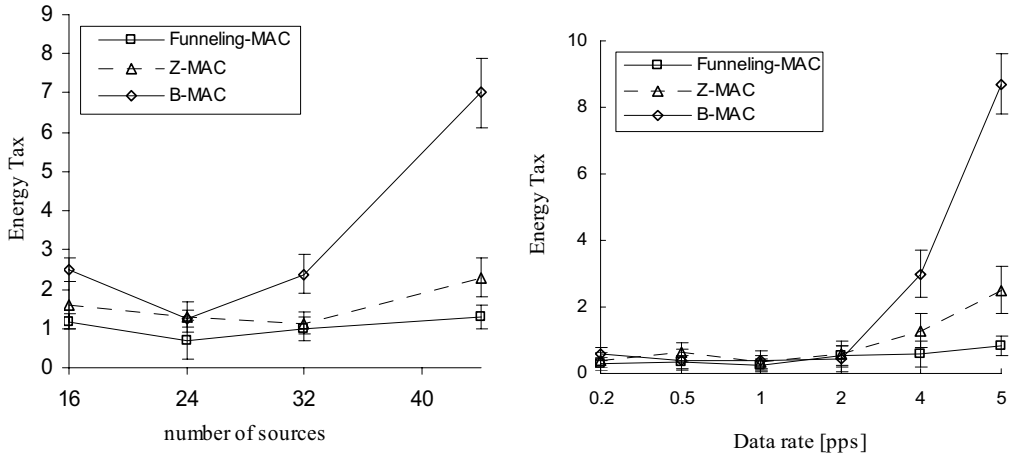
In order to analyze the cost of delivering data packets to the sink, we define the energy tax Etax as,

$$E_{tax} = \frac{D_t + C_t}{D_d \cdot n} \tag{3.10}$$

where D_t is the amount of data packets transmitted (by source nodes or intermediate nodes) in number of bits, C_t is the total amount of control packets transmitted in number of bits, D_d is the amount of packets delivered to the sink, and n is the number of nodes in the network. We measure the signaling overhead cost E_{sig} as,

$$E_{sig} = \frac{C_t}{D_d \cdot n}. \tag{3.11}$$

The energy tax includes the overhead of control messages if a MAC protocol introduces some control messages. The funneling-MAC introduces signaling for beacon packets, schedule packets, path information field, and meta-schedule, which we considered when computing the energy tax. B-MAC does not introduce any control packets. Z-MAC introduces sync packets for local synchronization. In Figures 3.20 and 3.21 we consider Z-MAC with and without the DRAND overhead. If a data packet has to travel i hops to reach the sink, the energy used in delivering this packet i hops is included in the energy tax as well. If a data packet is lost after traveling j hops, the energy used in delivering this packet j hops is included in the energy tax. The signaling overhead cost only considers the overhead of control messages and not the cost due to packet loss. The testbed settings are the same as the settings in Section 3.6.6. The funneling-MAC introduces more signaling overhead compared to B-MAC but the funneling-MAC reduces the energy wasted by reducing the packet losses to the extent that the funneling-MAC has lower or equal energy tax compared to B-MAC despite its signaling overhead, as we can see from Figure 3.20. B-MAC does not perform well when the data rate is greater than 2 pps resulting in a high energy tax for B-MAC. In



(a) Varying number of sources with 5pps (b) 44 sources with varying data rate

Figure 3.20: Energy tax comparison of the funneling-MAC, Z-MAC, and B-MAC.

contrast, the funneling-MAC exhibits consistently lower energy tax. Z-MAC also reduces the energy wasted by packet losses to the extent that Z-MAC has a lower or equal energy tax compared to B-MAC despite the overhead of sync packets. However, Z-MAC has a greater energy tax than the funneling-MAC when the data rate is greater than 2 pps. This is because the funneling effect impacts Z-MACs overall energy tax as packet loss increases. In summary, our results indicate that even though the funneling-MAC has more signaling in its basic protocol than the other protocols, it is a more energy efficient than Z-MAC and B-MAC.

The signaling overhead cost of the funneling-MAC and Z-MAC are similar if we do not consider the overhead of running DRAND. Also from the schedule drift exhibited in Figures 3.14, 3.15 and 3.16 for Z-MAC it would be necessary to re-run DRAND to boost its performance and resolve the schedule drift. Based on the observations from Figure 3.16, it would be costly to re-run DRAND every 12 minutes to maintain performance. If we do this then the cost of operating Z-MAC and its DRAND mechanism increases significantly, as indicated in Figure 3.21. Clearly, it is not tenable to re-run DRAND periodically because of the significant overhead incurred; this is clearly shown in Figure 3.21.

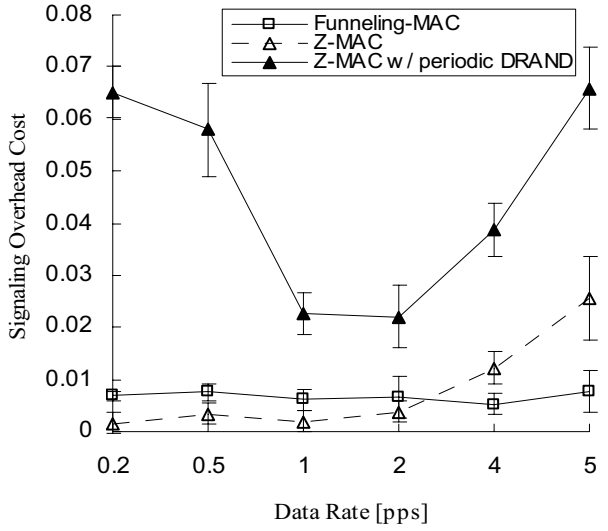


Figure 3.21: Signaling overhead cost of the funneling-MAC and Z-MAC.

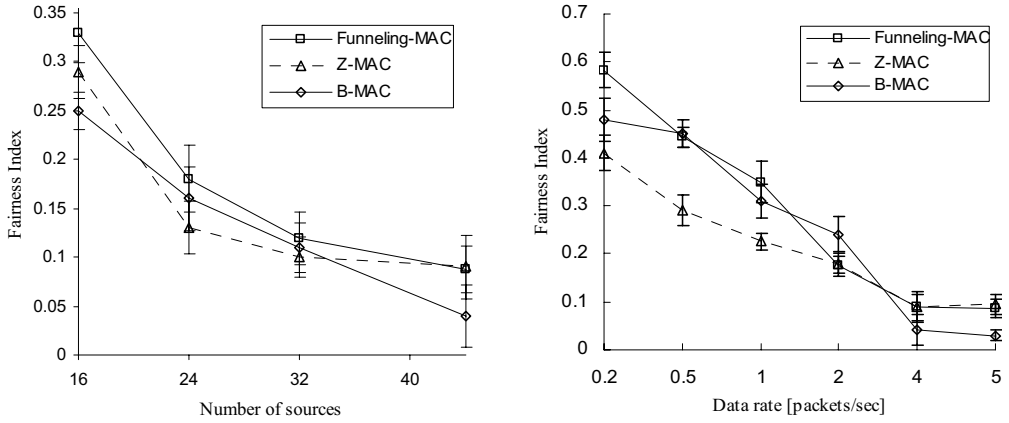
3.6.8 Fairness

To evaluate the fairness of the funneling-MAC, we measure the fairness index [49] of the throughput of the data packets from each source node as

$$f(x_1, x_2, x_3, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=0}^{n} x_i^2}. \tag{3.12}$$

where x_i is the throughput of the data packets from a source node i to the sink and n is the total number of source nodes in the network.

We evaluate the fairness index as a function of i) the number of sources in the network when each source transmits at 4 packets/sec (see Figure 3.22(a)), and ii) the per-source data rate when the number of sources is constant (see Figure 3.22(b)). Note that the distances from the sources to the sink vary from one hop to five hops. Given the lossy nature of wireless transmissions, it is natural that the probability of packet loss for a flow in a five-hop path is larger than the probability of packet loss for a flow in a one-hop path. Although the funneling-MAC improves the throughput as shown in Section 3.6.6, the improvement apply to all the sources including the nodes close to the sink and the nodes far away from the



(a) Varying number of sources with 4pps (b) 44 sources with varying data rate

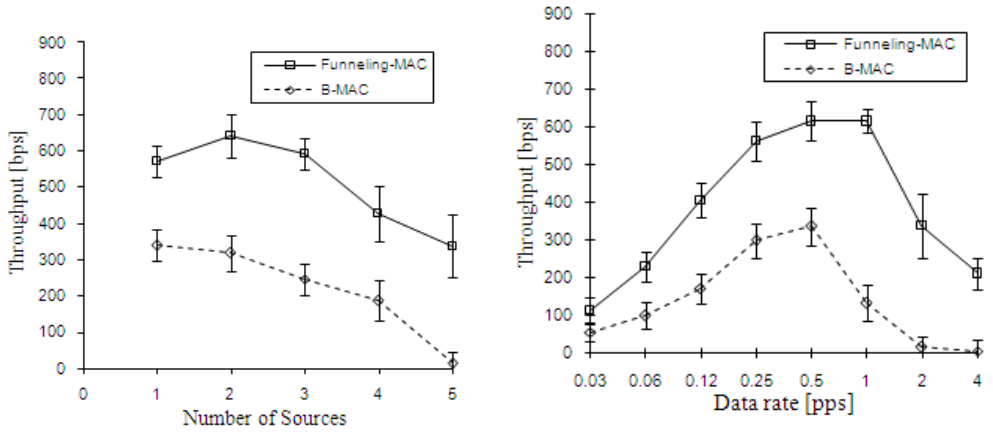
Figure 3.22: Fairness index of the funneling-MAC, Z-MAC, and B-MAC.

sink so it does not have much impact on the fairness. Hence it is not surprising that Figures 3.22(a) and 3.22(b) are presenting fairness lower than 0.5 fairness index in most of the cases for all three MAC protocols. This problem is addressed in [30] by introducing a fair congestion control algorithm at the transport layer arguing that sensor network applications may require minimum of data to be delivered from the most of the sources including the sources far away from the sink.

Figures 3.22(a) and 3.22(b) show that at high load the funneling-MAC and Z-MAC present better fairness than B-MAC. At low and medium load, the funneling-MAC provides similar or better fairness compared to Z-MAC and B-MAC. Combining the results in Sections 3.6.6, 3.6.7 and 3.6.8, it is shown that the funneling-MAC improves the performance in terms of throughput and energy efficiency compared to B-MAC and Z-MAC without sacrificing fairness.

3.6.9 Low Power Listening

This section presents the performance of the funneling-MAC when the low power listening (LPL) is enabled. The purpose of this experiment is to show that the funneling-MAC is



(a) Varying number of sources with 2 pps (b) 6 sources with varying data rate

Figure 3.23: The funneling-MAC and B-MAC Low Power Listening.

not conflicting with asynchronous duty cycling approaches such as the low power listening of B-MAC. This paper focuses on addressing the funneling effect and duty cycling is out of the scope of this paper. However, as we have discussed earlier in Section 3.4.6, it is necessary to ensure that the funneling-MAC can work together with duty cycling algorithms to improve the energy efficiency without any conflict. To show that, we use the multihop testbed in Columbia University and measure the throughput of the low power listening (LPL) of B-MAC and the funneling-MAC.

Each data packet uses the packet size of 36 bytes plus 8 bytes short preamble or 371 bytes long preamble. We set the duty cycle of 135 msec for the LPL of B-MAC and also for asynchronous LPL of the funneling-MAC. The f-nodes use the preamble size of 371 bytes and the slot size of 200 msec during the first TDMA frame after the beacon for the meta-schedule advertisement. The f-nodes use the preamble size of 8 bytes and the slot size of 30 msec in all other times. All other nodes use the preamble size of 371 bytes. We set the superframe size to be 5 seconds. The funneling-MAC dynamically assigns the size of TDMA, CSMA, and sleep frames as explained in Sections 3.4.3 and 3.4.6. We vary the number of sources (as shown in the x-axis of Figure 3.23(a)) and the packet generation

interval (as shown in the x-axis of Figure 3.23(a)). All other settings are the same as Section 3.6.1. The results are shown in Figures 3.23(a) and 3.23(b). The throughput of the B-MAC with LPL is significantly lower than the throughput of B-MAC without LPL shown in Section 3.6.6. The reason is clear because B-MAC with LPL use quite long preamble while the size of preamble is not counted as the throughput. Also, the probability of collision is higher with longer packet size (due to the longer preamble). In addition, the funneling effect reduces the throughput which is already low. As a result, the low throughput becomes a critical problem even for the applications with low duty cycle. The network breaks down (i.e., the throughput is almost zero) in such a low traffic load as 2 pps with 6 source nodes. The throughput of the funneling-MAC with LPL is more than twice as high as the throughput of the B-MAC with LPL as shown in Figures 3.23(a) and 3.23(b). The result indicates that the funneling-MAC can support LPL while mitigating the funneling effect under various traffic conditions.

3.7 Conclusion

The main contributions of this chapter are as follows. We show that by implementing a simple hybrid TDMA/CSMA scheme in the intensity region, under the control of the sink, can significantly improve the throughput and loss performance of sensor networks, even under lightly loaded traffic conditions, and for small intensity region depths of one or two hops. We also show experimentally that multiple MACs can coexist in the sensor network, specifically, we can run a hybrid TDMA/CSMA in the intensity region which seamlessly coexists with pure CSMA outside of that region, in addition, any potential interference caused by dynamically increasing or decreasing the intensity region (i.e., the TDMA/CSMA region) is effectively managed by the funneling-MAC.

We show through experimentation that the funneling-MAC outperforms B-MAC and its closest competitor Z-MAC under a wide variety of network and traffic conditions. The

TinyOS source code for the funneling-MAC is publicly available from the web [97].

Chapter 4

MetroTrack

4.1 Introduction

Event tracking is a key problem, important application, and active area of research in sensor networks. In the past tracking applications (e.g., surveillance, hazard tracking, and wildlife monitoring) have driven the deployment of sensor networks across a number of disparate domains, such as, battlefields, industrial facilities, and protected environmental areas. Urban sensing [17, 1, 27] is an emerging area of interest that presents a new set of challenges for traditional applications such as tracking noise, pollutants, air quality, objects and people (e.g., based on radio signatures using RFID tags [37]), cars, or, as recently discussed in the literature and popular press weapons of mass destruction [81].

When we think about the tracking problem, traditional solutions that come to mind are based on the deployment of static sensor networks. Building sensor networks for urban environments requires careful planning and deployment of possibly a very large number of sensors capable of offering sufficient coverage density for event detection and tracking. Static sensor solutions are also challenging because events are unpredictable in time and space. Therefore, unless the network provides complete coverage you have to determine in advance where to deploy your network. We believe the use of static networks across

urban areas has significant cost, scaling, coverage and performance issues that will limit their deployment.

An alternative design of such a sensor network, which we propose in this chapter, is to use mobile phones that people carry as mobile sensors to track mobile events. Increasingly, top-end phones are becoming more computation capable and embed sensors and communication support to make a sensor network built on mobile phones more of a reality. For example, many top end mobile phones such as Nokia N95 include a number of different radios (e.g., multiple cellular radios, WiFi, Bluetooth, and 802.15.4), sensors (e.g., accelerometers, microphone and camera, and GPS) that are programmable. We imagine that MEMS technology will allow for the integration of more specialized sensors (e.g., CO_2) in the future that is not possible today because of size, power, or form factor barriers.

There is a number of important challenges in building a mobile event tracking system using mobile sensors. First, mobile sensors need to be tasked before sensing [19, 109]. Another issue that complicates the design of the system is that the mobility of mobile phones (therefore, the mobile sensors) is uncontrolled. This work diverges from mobile sensing systems that use the controlled mobility of a device (e.g., a robot) as part of the overall sensor system design. In such cases the system can be optimized to drive the modality of the sensors in response to detected events [56]. Another insight is that the motion of people carrying mobile sensors (i.e., phones) is independent of the motion of the event to be tracked. More specifically, due to the uncontrolled mobility of the mobile sensors, there is no guarantee that there will always be high enough density of mobile sensors around any given event of interest. The density changes over time so that sometimes there is a sufficient number of devices around the event to be tracked and, at other times, there is limited device density. One can think of this as dynamic sensor network coverage. The event tracking process has to be designed assuming that the process of tracking will be periodically disrupted in response to dynamic density and, therefore, changing coverage conditions. Thus, a fundamental problem is how to recover a target when the system loses

track of the target due to changing coverage.

In our design, we assume that we can exploit the mobile phones belonging to people going about their daily lives or defined groups (e.g., airline employees, etc.). Ultimately, the more people opting in the better density and sensing coverage the more effective urban sensing systems will become in delivering services with better fidelity. In this chapter, we do not discuss what would incentivize more people to opt in nor do we discuss the important privacy, trust and security issues that predicate the wide scale adoption of these ideas. Rather, we leave those issues for future work and focus on a proof of concept, deployment and evaluation of a system capable of tracking mobile events using mobile phones, targeted to urban landscapes.

In this chapter, we propose *MetroTrack*, a system capable of tracking mobile events using off-the-shelf mobile phones. MetroTrack is predicated on the fact that a target will be lost during the tracking process and takes compensatory action to recover the target, allowing the tracking process to continue. In this sense, MetroTrack is designed to be responsive to changing density of mobile phones and changing sensor network coverage. The MetroTrack system is capable of tasking mobile sensors around a target event of interest and recovering lost targets by tasking other mobile sensors in close proximity of the lost target based on a prediction of its future location.

We implement the MetroTrack system using off-the-shelf Nokia N95 and N80 mobile phones. MetroTrack is based on two mechanisms, namely, *information-driven tasking* and *prediction-based recovery*. The tasking procedure is information-driven because each sensor node independently determines whether to forward the tracking task to its neighbors or not, according to its local sensor reading state. The solution is therefore fully distributed and uses local state only. If the sensor readings meet the criteria of the event being tracked then the sensor node forwards the task to its neighbors informing them it detected the event. The recovery procedure is based on prediction algorithms that estimate the lost target and its margin of error. MetroTrack adopts a geocast broadcasting scheme [59, 47] to forward the

task to the sensors in the projected area of the target. The predicted geographical position of an event is computed using a Distributed Kalman Filtering (DKF) estimation based on the control theory algorithm presented in [75]. MetroTrack tracks events based on local state and local radio (i.e., ad hoc WiFi) interaction between mobile phones in the vicinity of a target as a design goal and interacts with the back-end servers using cellular or infrastructure-based WiFi connectivity.

To the best of our knowledge, this is the first sensor based tracking system of mobile events using mobile phones. The contributions of this work are as follows:

- From a systems perspective, we present the design, implementation and evaluation of a tracking system completely based on off-the-shelf mobile phones.
- From an algorithmic perspective, we propose a novel distributed protocol for mobile sensors based on a consensus based distributed Kalman Filter algorithm for recovering lost targets and we analyze the algorithms performance by means of large-scale simulations using different mobility models and through the deployment of a real-world testbed using Nokia N80 and N95 phones.

This chapter is organized as follows. Section 4.2 describes MetroTrack's information-driven tracking and prediction-based recovery algorithms, and communication protocol. In Section 4.3, we present the mathematical formulation of the prediction algorithm that prediction-based recovery is founded on. In Section 4.4, we discuss the implementation of a prototype MetroTrack system, its deployment and performance evaluation. MetroTrack is implemented on Nokia N80 and N95 phones and as proof of concept is effective at tracking a mobile noise source in an outdoor urban environment. Following this, in Section 4.5, we study the large-scale design space of MetroTrack which can not be analyzed from a small scale testbed deployment. Through a set of comprehensive simulations that consider a number of different mobility models and deployment scenarios, we study the scaling and

performance properties of MetroTrack. Simulation results indicate that MetroTrack is robust in the presence of different mobility models and mobile density. In Section 4.6, we discuss the related work and conclude in Section 4.7 with some final remarks.

4.2 MetroTrack Design

In this section, we present the MetroTrack distributed tracking algorithm which includes the coordination mechanisms between mobile devices. We discuss tracking initiation, information-driven tasking, and prediction based recovery.

4.2.1 Distributed Tracking Algorithm Design

First, we start with an overview of the design of the tracking algorithm. The states and the state transitions are briefly described in this sub-section. More details of the states and the states transitions are explained from Section 4.2.2 to Section 4.2.4.

Device States

MetroTrack is able to track an event by progressively tasking devices that are in its vicinity and recover if the event is temporarily lost by means of a distributed target recovery process. A device running the MetroTrack system can be in any one of four states (as shown in Figure 4.1), which correspond to the four phases of the tracking process itself; these are:

- **Idle**: In this state the device is not currently performing any action of the tracking process.
- **Tasked**: A device enters this state after receiving a request (a task) to track a certain event. The request contains a description of the event (e.g., the description of a particular sound signature to be detected in the case of our implementation).

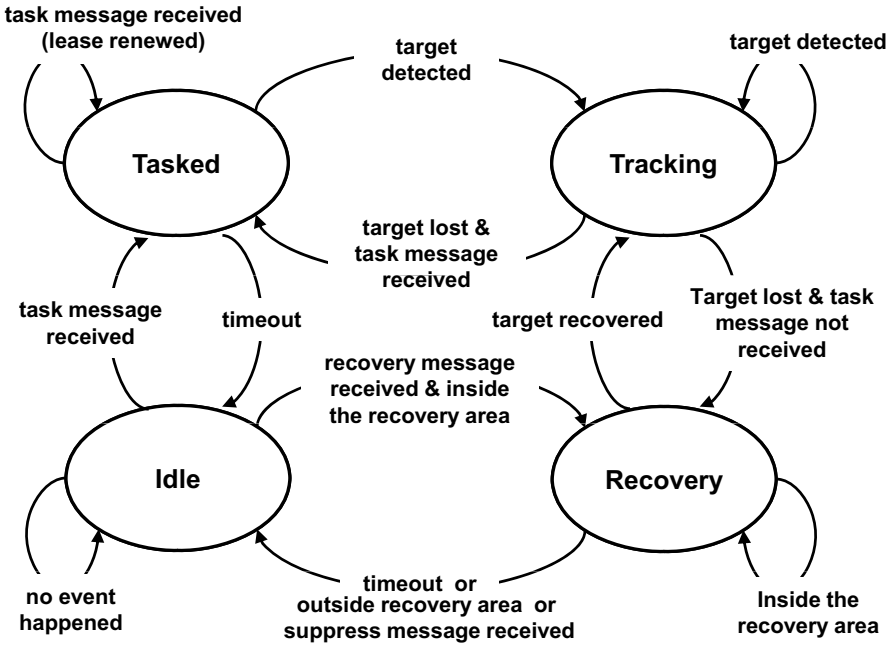


Figure 4.1: State diagram describing the MetroTrack tracking process.

- **Tracking**: When a device finds the target, it starts tracking it. A device tracking an event sends task messages periodically to all its neighbors.

- **Recovery**: While tracking a target, a device might lose the target. In this case, the device will initiate a recovery process. The device sends a recovery message to all its neighbors in order to start a recovery process. Prediction techniques are used to identify a recovery area where the target is predicted to be found. All the nodes in that area attempt to recover the target.

State Transitions

In what follows, we provide an overview of the tracking process, describing how and when state transitions occur.

- a device is initially in the *idle* state; it can start a tracking process by sending a task message to its neighbors; this message should be periodically re-sent in order to

continue the tracking process;

- when an *idle* device receives a task message, it becomes *tasked*; a lease duration (i.e., soft state) is associated with the task message received (i.e., the device will be in *tasked* state for certain time interval defined by the lease timer). The lease is renewed if a new task message is received. If neither the reception of a new task message nor event detection takes place, the device returns to the *idle* state;

- when a device discovers an event, it enters the *tracking* state;

- a device in the *tracking* state returns to the *tasked* state if it loses the target and receives a task message; however, if it loses the target and does not receive a new task message it enters the *recovery* state, i.e., the device attempts to localize and track the target again;

- if a device in the *recovery* state recovers the target, it returns to the *tracking* state. Alternatively, if it is not able to recover the target it returns to the *idle* state. A device in the *idle* state can also enter the recovery phase if it receives the recovery message and is inside the projected recovery area. A device can also be tasked again if it receives the task message (i.e., it moves to the *tasked* state).

We now describe the details of the tasking and tracking processes.

4.2.2 Tracking Initiation

The tracking of an event can be initiated in two ways: user initiation or sentry sensor [19] initiation. A user can request to track an event described by certain attributes when the target event is encountered. For example, when a user encounters a noise maker in the neighborhood, the user can record the noise signature and request to track the source of the noise. Another way is to rely on sentry nodes to detect the event to be tracked. The sentry

nodes can be selected from mobile nodes that have enough power to periodically turn on their sensors and start sampling. When one of the sentry nodes detects an event that matches the pre-defined event description, the node initiates the tracking procedure. For example, the event descriptor of a loud sound to be tracked if the root mean square is greater than 200 can be defined as follows (using an XML-like notation):

```
<eventDescriptor id="0414200800001">
  <eventType>noise</eventType>
  <sensorType>microphone</sensorType>
  <threshold>200</threshold>
</eventDescriptor>
```

If there are static sensor nodes deployed in the area (e.g., noise sensors on street lights) they can also take the role of sentry nodes. When one of the static sensors detects the event, it will send a task message to the mobile sensors in the area. The communication between mobile sensors and static sensors can be established using local radio such as Wi-Fi, 802.15.4, or Bluetooth.

The task message specifies the characteristics and the attributes of the events to be tracked. The attributes of the event include the modality of sensors that can detect the event (i.e., the type of the sensor or set of sensors that are able to detect the event) and the threshold value to use for event detection. Sampled data can be pre-processed for example by means of low-pass, high-pass, or band-pass filters. In many applications, such as, in the case of the detection of a specific sound wave (or, more generally, an energy wave) the detection can be simply based on the fact that a characteristic of the current sensor reading is greater than a threshold value (e.g., the amplitude of the wave is higher than a given threshold). For other types of events the detection can be determined by a combination of conditions and thresholds. A possible solution for event matching is the adoption of a light-weight Likelihood Ratio Test [65].

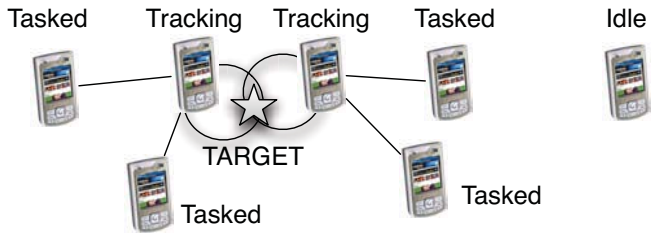


Figure 4.2: Information-driven Tasking.

The device associated with the requesting user or first sentry node that has detected the event becomes an initiator. Each neighboring sensor node that receives the task message performs sensing. The sensor node does not forward the task message to its neighbors unless it detects the event.

4.2.3 Information-driven Tasking

The task message is forwarded by the sensors that are tasked and have detected the event, as shown in Figure 4.2. These forwarding sensors replicate the task message and broadcast the task message to their neighbors. As a result, the nodes in close proximity to the event are tasked and the size of the tasked region is one hop wider than the event sensing range.

The tasking process is composed of a reactive mechanism and a proactive mechanism. The tasking process can be seen as reactive because the task message is forwarded after the detection of the event. At the same time, it can also be considered proactive since the sensors just outside (within one hop) the event sensing range are already tasked and ready to detect the event wherever it moves. A sensor node determines whether it has detected the event by comparing the sensor reading (e.g., using the microphone) and the description of the event in the task message. As discussed earlier, the description of the event includes the modality of the sensors that can detect the event and the methodology by which the event can be detected (such as a threshold value). If the modality of the sensor node matches one of the modalities specified in the task message (i.e., the device is able to sense the event),

then the sensor node starts the sampling process.

The responsibilities of the sensor that detects the event are as follows. The sensor should keep sensing the event using a high sampling rate and report the data to the back-end servers. In addition, the sensor should periodically forward the task message to its neighboring sensor nodes. The sensors that are tasked with one of the task messages containing the same event identifier form a tracking group. We note that the information driven tasking is not based on the election of a leader. Maintaining a leader for a group requires overhead. In addition, the failure of the leader affects the overall operation of the tracking system. MetroTrack can maintain the group and task the sensors to track the target without the need of a leader.

4.2.4 Prediction-based Recovery

Due to uncontrolled mobility of people carrying mobile phones and the varying density of people passing through an specific area it may happen that there will not be sufficient mobile sensors to detect and track the target over time. Therefore, the target event is likely to be lost from time to time. If the target moves to a new location resulting in the tasked sensors losing the target then untasked sensors in the new area cannot detect the target even though the target is within their sensing range. For this reason, MetroTrack performs a prediction-based recovery process to enhance the probability of quickly detecting the target again. This is an important component of the overall protocol that makes the system effective under conditions where targets are lost from time to time due to the potential sparseness of the network.

Recovery Initiation

The task of tracking the event is distributed among multiple mobile sensors. If a sensor is not detecting the event, this is not considered sufficient to infer that the target is completely lost

since other sensors may still be sensing the event. In MetroTrack, mobile sensors listen to the recovery message from their neighbors to minimize the false positives of such decision. A sensor that has detected the event previously but currently is not detecting the event listens to the task messages forwarded from its neighboring nodes. If none of the neighboring nodes are forwarding the task message, the device infers that the target is lost. Then, the node initiates the recovery process.

Assuming that the speed of the target is comparable to that of a tracking node and the sampling rate of sensors is high enough to detect the event, the overhearing will prevent false positives. However, there might still be false positives if the density of sensors is not sufficient. If a sensor makes a wrong decision, each node will forward an unnecessary number of task requests. However, the penalty is bounded by limiting the duration of the recovery process.

Recovery Process

When one of the sensors declares that the target is lost as described above then the sensor initiates the recovery process by broadcasting a recovery message. The recovery process is based on the estimation of the location of the lost target and the error margin associated to the prediction. The recovery message contains the information about the lost target. MetroTrack adopts a geocast broadcasting scheme [59, 47] to forward the recovery message to the sensors in the projected area where the target will likely move to (based on prediction). The sensors that receive the recovery message attempt to detect the target. If one of the sensors receiving the recovery message detects the target then the recovery process is complete. The sensor that recovered the target broadcasts a task message, which resumes the information-driven tasking part of the protocol. As illustrated in Figure 4.3, all the hosts in the recovery area are in the recovery state.

We considered a projected circular area. The center of the projected area is the predicted

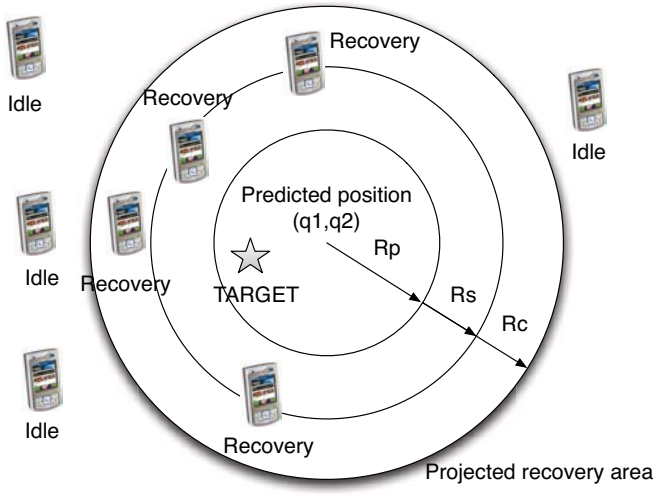


Figure 4.3: Prediction-based recovery.

target location and the radius is the error margin of the prediction. The calculation of this area is based on the application of Kalman filter based forecasting techniques as described in Section 4.3.

The radius R of the recovery area that MetroTrack uses is calculated as sum of the error margin of the prediction, the sensing range and the communication range:

$$R = R_p + R_s + R_c \tag{4.1}$$

R_p corresponds to the error margin associated to the prediction (see Equation 4.8 in Section 4.3.1). Moreover, our goal is to task all the sensors that are likely to be in contact with the target inside the projected region so we add the sensing range (R_s) to this radius. Finally, we also add the communication range of the devices (R_c) in order to be able to have the nodes that are at one-hop distance from those at the border of the area with radius R_p+R_s in recovery state. These nodes are likely to enter the area and are particularly useful in spreading the recovery messages in the case of sparse network topologies. We note that the disk-shape model is an approximated conceptual model that in a real deployment is influenced by the GPS errors for localization and by non uniform radio propagation and

interferences. We also implement MetroTrack in a real testbed and study the performance of the tracking algorithm.

A node that has received the recovery message stays in recovery state until the node moves outside the recovery area or the recovery process timer expires. A timeout is specified to limit the duration of the recovery process. If the target is not recovered after the expiration of the timer, MetroTrack stops tracking the target. The nodes in recovery state broadcast the recovery message to their instant (one-hop) neighbors periodically so that new nodes that move into the recovery area can receive the recovery message. This solution is similar to the use of forwarding zone in geocast [59, 47] schemes.

Suppression Process

It may happen that some sensors can be still in the recovery state while other sensors have already recovered the target and started to track it. It may happen that the target event disappears (e.g., a sound that is suddenly silent). MetroTrack addresses this problem using two mechanisms. First, it limits the duration of the recovery process and the spatial dissemination of the recovery messages. Second, MetroTrack performs a *suppression process* to reduce unnecessary overhead. Every node that recovers the target or receives a task message, broadcasts a suppression message that is disseminated among the devices in the recovery area. Every node that receives the suppression message inside the recovery area re-broadcasts the message, or, if the node is in recovery state, it stops the recovery process and stops broadcasting the recovery message. Although the suppression message might not reach every node in the recovery state, the suppression process reduces the message overhead considerably as we show in Section 4.5.6.

4.3 Distributed Kalman Filter Prediction Algorithm

In this section, we present the prediction algorithm that we implemented in MetroTrack for supporting the recovery process based on the Distributed Kalman Consensus Filter described in [75]. We provide an overview of the prediction model that we use for the calculation of the projected area in order to fully understand the collaborative prediction protocol used for the recovery process.

4.3.1 Prediction Model

In what follows, we define a generic model for predicting the movement of a target in geographical space. We consider a moving target with position $q \in \Re^2$ and a constant velocity $p \in \Re^2$. The assumption of constant velocity is acceptable for deployment in urban areas where the variance of the average speed of a node is limited. The one-step predictor is defined as follows:

$$\hat{x}(k+1) = A\bar{x}(k) + Bw(k) \tag{4.2}$$

where $x(k) = [q_1(k), p_1(k), q_2(k), p_2(k)]$ denotes the state of the target at time k. $\bar{x}(k)$ indicates the *prior state estimate* at step k given the knowledge of the movement under observation, whereas $\hat{x}$ indicates the *state estimate* of the same process at time $k+1$. q_1 and p_1 are the position and the speed on the x-axis and q_2 and p_2 are the position and speed on the y-axis, respectively. $w(k)$ is a zero-mean Gaussian noise denoted by $N(0, 1)$. The prior estimate is the information stored in the phones and periodically exchanged among the phones that are in reach. The matrix A and B are defined as follows:

$$A = \begin{pmatrix} 1 & \epsilon & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \epsilon \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$B = I_2 \otimes G$$

with

$$G = \begin{pmatrix} \epsilon^2\sigma_0/2 \\ \epsilon\sigma_0 \end{pmatrix}$$

and $\otimes$ denotes the Kronecker product of matrices.

The prediction for the instant $k + 2$ is defined as follows:

$$\hat{x}(k+2) = A^2\bar{x}(k) + ABw(k) + Bw(k+1) \tag{4.3}$$

The generic prediction for the instant $k + m$ is defined as:

$$\hat{x}(k+m) = A^m\bar{x}(k) + \sum_{j=0}^{m-1} A^j Bw(k+m-1-j) \tag{4.4}$$

The meaning of the symbols $\hat{x}$ and $\bar{x}$ is the same of the $k + 1$ case. This equation can be rewritten as:

$$\hat{x}(k+m) = A^m\bar{x}(k) + v(k) \tag{4.5}$$

where $v(k)$ is the noise associated to the $k + m$ prediction defined as:

$$v(k) = \sum_{j=0}^{m-1} A^j Bw(k+m-1-j) \tag{4.6}$$

The variance of $v(k)$ is

$$R_v = \begin{pmatrix} \sigma^2_{v_{q_1}} & 0 & 0 & 0 \\ 0 & \sigma^2_{v_{p_1}} & 0 & 0 \\ 0 & 0 & \sigma^2_{v_{q_2}} & 0 \\ 0 & 0 & 0 & \sigma^2_{v_{p_2}} \end{pmatrix}$$

$$= [\sum_{j=0}^{m-1} A^j BB^T(A^j)^T]R_w \tag{4.7}$$

where

$$R_w = I_4$$

Therefore, the center of the recovery region is $(\hat{q}_{(k+m)_1}, \hat{q}_{(k+m)_2})$. We consider a radius for the recovery area equal to:

$$r = max[2\sigma_{v_{q_1}}, 2\sigma_{v_{q_2}}] \tag{4.8}$$

The value is r is chosen in order to obtain a 95% confidence interval for the projected recovery area. In other words, we can assume that the target will be located in the recovery with approximately 95% probability.

4.3.2 Distributed Kalman Filter

We use the Kalman-Consensus filter presented in [75] as a basis of the distributed prediction of the location of the target for the recovery process. Each node i runs the distributed estimation algorithm showed in Algorithm 1. We indicate with z_i the observation performed by each node. N_i indicates the neighbors of node i. The message that is periodically broadcasted contains the following tuple:

$$msg_i = [u_i, U_i, \hat{x}_i]$$

The local aggregation and calculation is described in step 3, whereas the estimation of the consensus among the neighbors is performed in step 4. The equations of the update of the filter are presented in step 5.

The sensing model that we use is the following:

$$z_i(k) = H_i(k)x(k) + v_i(k) \tag{4.9}$$

where $H_i(k)$ is the observation matrix and $v_i(k)$ is the zero-mean Gaussian noise of the

measurements of the ith node with covariance R_i. In our implementation, we assume that the value of the observation matrices $H_i(k)$ is the same for the all nodes over time and it is equal to:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

We also assume that the value of R_i is equal to a constant for all the matrices:

$$R_i = \sigma_R^2 I_2 \tag{4.10}$$

The value of Q is the same for all the devices since it is only dependent on the value of the process under observation that is the same for all the devices (i.e., the position of the moving target):

$$Q = \sigma_0^2 I_4 \tag{4.11}$$

Finally, P_0 is defined as

$$P_0 = \sigma_R^2 I_4 \tag{4.12}$$

4.4 Implementation and Experimental Evaluation

We built a proof-of-concept mobile phone based testbed to evaluate the MetroTrack system. The idea is to deploy a real system trying to re-create a practical mobile event tracking situation in a urban area given the constraints in terms of number of available mobile phones and people carrying the devices. The testbed consists of Nokia N80 and N95 smart phones (see Figure 4.4(a)), running Symbian OS S60. Both of them are equipped with a microphone and a camera that are accessible via software. With respect to the network connectivity they are both equipped with Bluetooth and WiFi interfaces. N95 also features integrated GPS and

Algorithm 1 Distributed Kalman Filter algorithm.

1: Initialization: $P_i = P_0, \bar{x}_i = x(0)$
2: **while** new data exists **do**
3: Locally aggregate data and covariance matrices:

$$J_i = N_i \cup \{i\}$$
$$u_j = H_j^T R_j^{-1} z_j, \ \forall j \in J_i, \ y_i = \sum_{j \in J_i} u_j$$
$$U_j = H_j^T R_j^{-1} H_j, \ \forall j \in J_i, \ S_i = \sum_{j \in J_i} U_j$$

4: Compute the Kalman-Consensus estimate:

$$M_i = (P_i^{-1} + S_i)^{-1}$$
$$\hat{x}_i = \bar{x}_i + M_i(y_i - S_i \bar{x}_i) + \epsilon M_i \sum_{j \in N_i} (\bar{x}_j - \bar{x}_i)$$

5: Update the state of the Kalman-Consensus filter:

$$P_i \leftarrow A M_i A^T + BQB^T$$
$$\bar{x}_i \leftarrow A\hat{x}_i$$

6: **end while**

an accelerometer. Since N80 phones are not equipped with GPS, we used an external dongle (see Figure 4.4(a)) based on the SiRFstar III chipset connected to the phone via Bluetooth. The devices uses GPS information for sound source localization as well as geocasting of recovery message.

4.4.1 Implementation

In our testbed, we use WiFi for local ad hoc communications between mobile phones and adopt UDP broadcasting; each device listens to three predefined ports, one for task and recovery messages, one for location information, and one for the messages related to the Distributed Kalman filter. The MetroTrack system is written in Python and is based on PyS60 (Python for S60) [74], Nokia's porting of Python 2.2 for Symbian OS S60. PyS60

(a) From left to right: N95, GPS dongle, N80. (b) Boombox bike used as mobile sound target.

Figure 4.4: Testbed devices.

supports the standard features of Python as well as several specific modules to access the phone functions and the onboard sensors (e.g., camera, microphone, accelerometer and GPS). Developers can easily extend it to access the native Symbian API and programs using the C/C++ extension module. Currently, Pys60 is more flexible than the Nokia implementation of J2ME for the N80 and N95 phones with respect to the programming interface for accessing the sensors embedded on the phones.

4.4.2 Testbed Setup and Experiments

Our experiments consist of tracking a mobile noise source; more specifically the boombox bike shown in Figure 4.4(b) that plays constant pink noise (i.e., a signal with a frequency spectrum such that the power spectral density is proportional to the reciprocal of the frequency). To perform this task, we implement an experimental sound source tracking application interfaced with the MetroTrack system. The system architecture is illustrated in Figure 4.5. The goal of this set of experiments is twofold: first, testing and evaluating the performance of the MetroTrack algorithms; second, evaluating the interfacing between the sensing system (in this case, the sound detector), MetroTrack, and the associated localization scheme based on triangulation, also considering its limitations and practical implementation

issues. We note that MetroTrack is independent of the underlying localization scheme.

We record sound samples using the microphone every 2 s. Similarly to [65], to estimate the distance from the target we compute the Root Mean Square (RMS) of the average sound signal amplitude. If the calculated RMS value is distinctively greater than the background noise, the sensor determines that the target event is detected and feeds the RMS value to the distance estimation component. An alternative method is *bearing estimation* [40], but it is not applicable to mobile phones since it requires at least two microphones on one device with known orientation: for this reason, it is not suitable for an opportunistic solution like ours where the users do not take an active role in the sensing process. We note that our sound source localization is a limited prototype system. For better accuracy, Discrete Fourier Transform (DFT) Filter or Fast Fourier Transform (FFT) Filter can be used to eliminate the background noise.

We implement two prediction mechanisms, a simple Local Kalman filter and a consensus-based Distributed Kalman filter in order to evaluate the trade-offs between the two. The Local Kalman filter is simply a special case of the Distributed Kalman filter [54] without any sharing of information among the devices about the position of the target. We implement the Distributed Kalman filter as described in Section 4.3. With respect to the mathematical model presented in that section for the Local Kalman filter we assume that $J_i = \emptyset \cup \{i\}$.

The distance between the sensor and the sound source can be estimated from the RMS value assuming that we know the original volume of the target sound and the pattern of the sound attenuation over distance. The prototype is based on a 2-D triangular localization scheme [110]. After estimating its location and distance from the target, each sensor shares this information with its one hop neighbors. If the sensor receives the information from more than two neighbors, it tries to estimate the target location using the triangular localization scheme. If the sensor has instead received the information from only one neighbor, each host estimates the target location by calculating the weighted average between its location and the location of the neighbor using the distance of the other node from the target over the

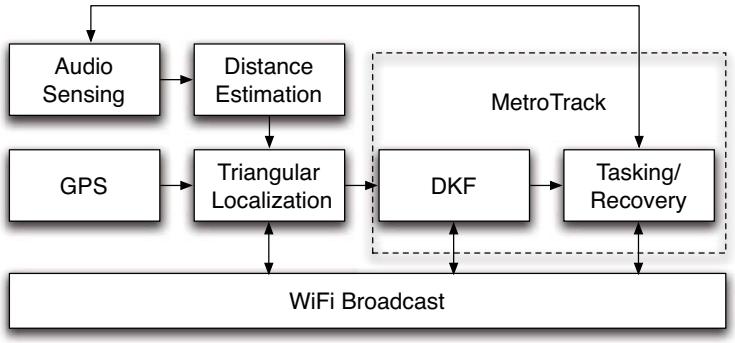


Figure 4.5: System Architecture.

sum of the two distances as weight (i.e, more importance is given to the estimation provided by the mobile that is closer to the target). The target location estimated by the sound source localization is fed into the Distributed Kalman filter component as the observation (z_i) of the node i (see Section 4.3).

We setup a testbed composed of two N95 phones and nine N80 phones connected to nine Bluetooth GPS dongles. We mount the boombox on the back of a bike (aka boombox bike) and we cycle it at a slow pace along paths around a university campus at approximately walking speed. We set the speaker of the boombox to face down toward the ground (as shown in Figure 4.4(b)) so that the sound is reflected and spreads omni-directionally in 2-D dimensions. The sound is sampled by the microphone on the phones for 0.5 s. The sampling is performed every 2 s. The time interval between each sampling is 1.5 s. Because the mobile phones are not always performing the tracking process (i.e., it can be defined as opportunistic), we argue that the maximum achievable sampling rate and minimum transmission interval of the messages should be used. Energy cost is not an issue if the device is not often involved in the tracking process. However, even if it is not in general critical also given the design goal of this class of systems, we evaluate this performance issue of MetroTrack in Section 4.5.6. The values of the intervals are those sufficient for both the N80 and N95 phones for the RMS calculation, the distance estimation, the sound source localization, and the Distributed Kalman filter update calculation. We note that in

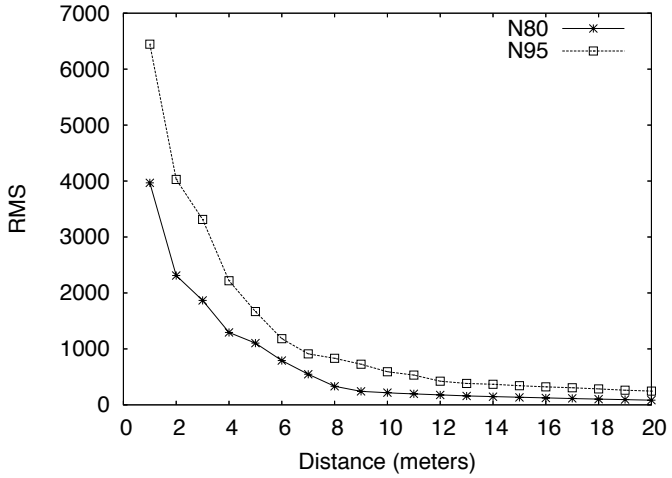


Figure 4.6: RMS measured vs. distance from the target.

existing tracking systems, the time intervals are much smaller than those used in MetroTrack (i.e., approximately 0.1-0.2 s) [43]. We set the WiFi transmission power to 100 mW. The communication range is between 25 and 30 m.

4.4.3 Experimental Results

First, we measure the RMS value of the sound sampled from 1 m to 20 m distance. The result is shown in Figure 4.6. It is interesting to note that the attenuation functions of the N80 and N95 phones are different. We perform the experiments during a quiet time around campus; the RMS value of the background noise in the area never reached a value above 240 for the N95 phones nor 80 for the N80 phones. We use these RMS values as thresholds for detecting the target. If the RMS value is above the threshold, the sensor determines that the target is detected. The distance is estimated from the measured RMS value using the curve shown in Figure 4.6 as conversion function. Given the threshold and the conversion function, we determine the sensing range is 20 m.

Then we perform the sound source tracking experiment evaluating the accuracy of the sound source localization as well as the effectiveness of the MetroTrack tasking and recovery. The GPS trace of the target is shown in Figure 4.7. Each person carries a phone and a

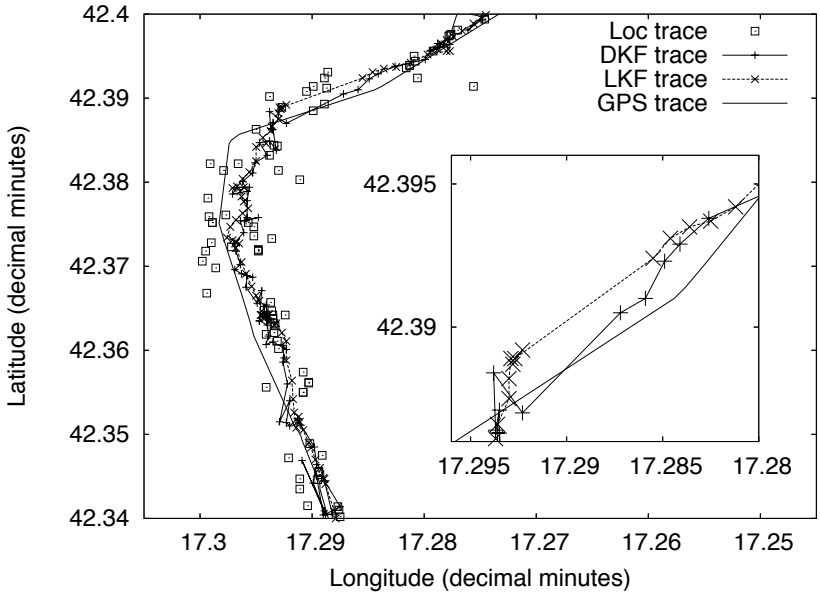


Figure 4.7: Trace of target location of the experiment.

Bluetooth dongle. Given the limitation of the number of phones and people, we emulate the density of a urban setting by allowing people to move around within 40 *m* from the target (i.e., the boombox bike). Given the restriction of being within 40 *m* from the target, each person was allowed to move randomly in and out of the sensing range (that, as we said, is approximately 20 *m*). This mobility setup is sufficient for testing the effectiveness of tasking process. We emulate the case of losing the target by turning the sound off for 16 *s* and then turning the sound on again to observe whether the recovery process is working effectively.

The trace of the target measured using the sound source localization scheme is shown in Figure 4.7 (see the curve *Loc trace* in the plot). As we can observe in the figure the measured location is noisy. The sound source localization error is not only caused by the error of the RMS measurement but also by the error of the GPS positioning estimation of the mobile sensors. Each mobile sensor uses its own GPS receiver and the accuracy of these receivers also varies, even if they are of the same model. Moreover, some mobile sensors do not have valid GPS readings at all on a cloudy day. We learned that calibrating the

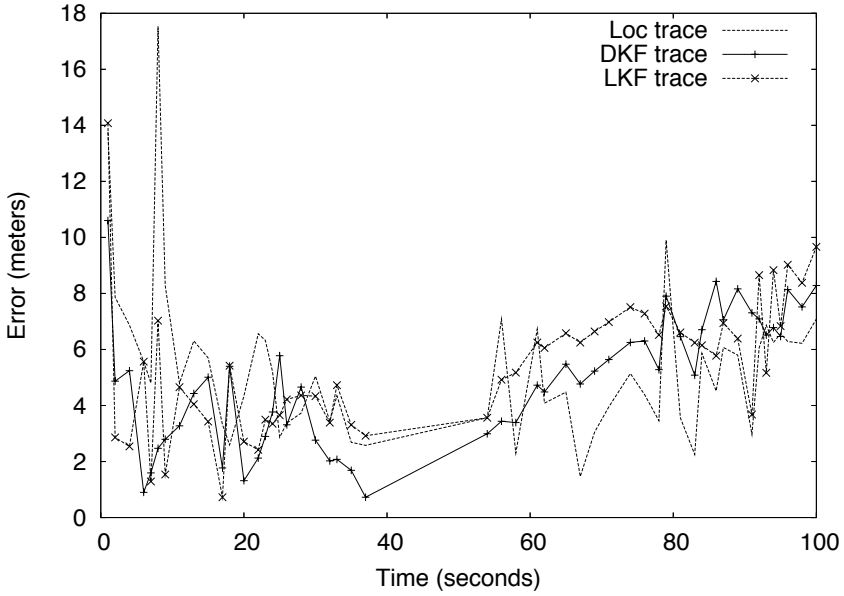


Figure 4.8: Time trace of the error of the target location estimation.

GPS reading among different sensors and checking the integrity of the GPS position of the mobile sensors is a real challenge that needs to be addressed in the future. The inset shows a zoomed section of the gap in the traces related to the recovery phase. The localization traces are not shown in the inset for clarity.

We also test the Local and Distributed Kalman filters estimations setting σ_R to 7 m because we learned through several trials of the experiments that the standard deviation of the sound source localization error (σ_R) is approximately 7 m. The trace of the Local and Distributed Kalman filters estimations of the target location is also shown in Figure 4.7. In order to evaluate the correctness of the prediction mechanism, we plot the distribution of the error of the location estimation in Figure 4.9. Figure 4.8 shows the time evolution of the estimation error referred to the traces depicted in Figure 4.7. The target in this picture starts at instant t =0 from the top of the area to the bottom. In Figure 4.8 we show the time interval of the first 100 seconds including the interval during which the target was lost (i.e., between time t=37 s and t=54 s). We observe that the Distributed Kalman filter estimation error is always small for this scenario, particularly given the localization error.

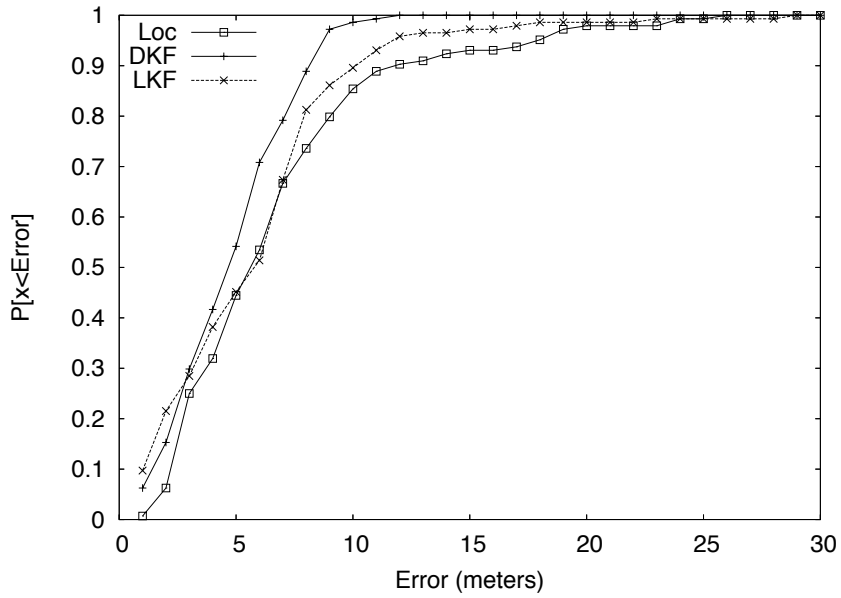


Figure 4.9: Cumulative distribution function of the error.

We would like to note that the goal of this experiment is not to test the effectiveness of the calculation of the recovery area given the limited number of devices. For this reason, we run large-scale simulations concerning this aspect of the system (see Section 4.5.5). The goal of this experiment rather is to test the correctness of the system in terms of transitions between the different tracking phases and the prediction mechanism itself also considering the limitations of the sensing devices and the error of the localization algorithm based on triangulation.

We perform benchmark tests on the battery life of N80 and N95 phones. The battery capacities of N80 and N95 are 890 *mAh* and 1200 *mAh* respectively. The results are presented in Table 4.1. We use three N95 phones and three N80 phones. These phones are used to perform the sound source tracking. We modify the testbed code to setup two sets of benchmark test. In the first benchmark, the phones are always tasked without detecting the target. Hence, these phones are passively sampling the noise with their microphone every 2 *s* and they do not broadcast any message. In the second benchmark, the phones are always tasked and consistently track the target. We feed the phones with an emulated target

location. These phones are programmed to sample the noise, send out their location for triangulation and the Distributed Kalman filter messages every 2 s. We recharge the battery of these phones before each test and measure the time difference between the starting time and the time that the phone is automatically turned off due to draining of the battery. The GSM interface is on in all the experiments. The experiments are performed in the same location with approximately the same GSM signal strength. We calculate the average of multiple experiments using the same type of phones. The results in Table 4.1 show that even the passive sensing is a heavy task that drains the battery in less than 3 hours and 40 minutes. The sensing and tasking processes drain the battery of the phones in less than 3 hours. This result supports the key idea of MetroTrack of tasking the mobile phones just in time only when the target is in the vicinity of the phones.

4.5 Simulation Study

We evaluate the performance, robustness, and scalability of MetroTrack for a number of different deployment scenarios using the Matlab simulator. The simulation results complement the experimental evaluation by studying issuses not easily evaluated in a small scale testbed, such as, scaling, and a sensitivity analysis of the system in the presence of different mobility models. We also evaluate the comparative performance of the Distributed Kalman filter and the Local Kalman filter. The parameters used during the simulation are shown in Table 4.2.

We run each simulation scenario 20 times with a simulation duration of 300 s for each scenario. The target event is active from the beginning to the end of every simulation run.

Table 4.1: Battery life benchmark (case of devices always involved in the tracking process).

Tasks	**N80**	**N95**
Sensing	$3h28m$	$3h14m$
Sensing + Tracking	$2h52m$	$2h47m$

The simulation area is a 1000 m by 1000 m square. We assume omni-directional radio model with transmission range of 100 m. The sensing range is 100 m for most scenarios except for the simulations whose results are shown in Figures 4.14(b) and 4.14(c). Every mobile sensor is characterized by a particular sensing range; this is also affected by the type of target we are tracking. We select a value representing a specific technology and we study the impact of the value of the sensing range in Figures 4.14(b) and 4.14(c). If the target is within the sensing range of a tasked sensor, the sensor is able to estimate the location of the target. An error is associated with the estimate. The distribution of the error is modeled using a zero-mean Gaussian distribution with standard deviation σ_R. Targets characterized by mobility patterns with larger standard deviations are more difficult to track. We study the impact of σ_R in Section 4.5.2. The sampling period (ϵ) is 1 s. For every mobile sensor tasked, it estimates the location of the target every second. The timeout value for the recovery process is 20 s.

We consider three mobility models for the target and the mobile sensors, namely the Constant Velocity model (also called Random Direction model with constant speed) [15], the Random Way-point model [50] and the Manhattan model [6]. The Constant Velocity model is the underlying model that MetroTrack uses for the Kalman filters, as discussed in Section 4.3. When a target or sensor node reaches the boundary of the simulation area, it

Table 4.2: Simulation parameters.

Parameters	**Default value**
Simulation area	1000 $m \times$ 1000 m
Simulation time	300 s
Number of hosts	144
Sensing range	100 m
Communication range	100 m
Communication error	10%
σ_R	20 m
σ_0	0.2 m/s
Sampling interval (ϵ)	1 s
Timeout recovery process	20 s

changes its direction randomly choosing a direction inside the simulation area with a goal toward the direction of one of the others sides of the simulation area. The standard deviation of the movement dynamics of the target and sensor nodes σ_0 is 0.2 m/s.

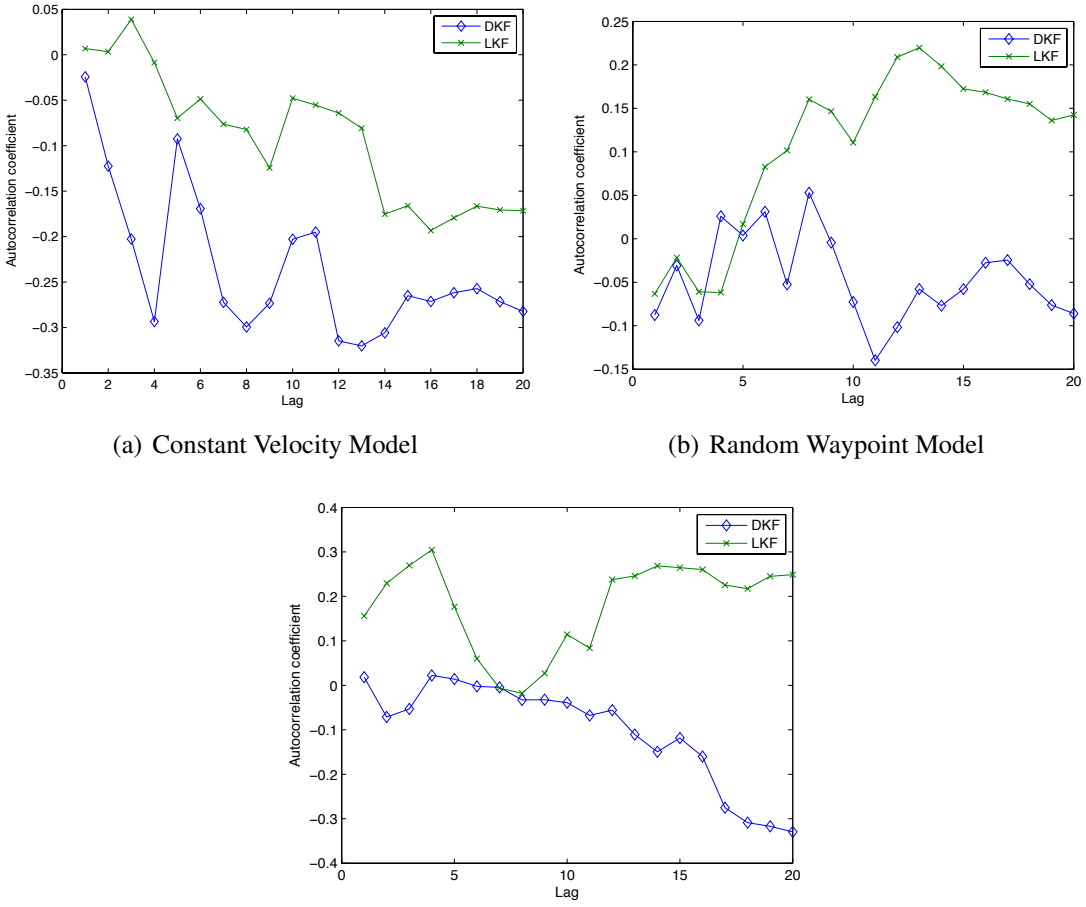
The Random Way-point model is a well-known mobility model used by the wireless networking research community. According to this model, the target and sensor nodes choose a destination point inside the simulation area and move toward the target at a constant velocity. The point is selected using a uniform random distribution. The speed is generated by means of a Gaussian distribution with mean equal to 1 m/s and standard deviation equal to 0.2 m/s.

We also study the impact of using the Manhattan model where the target and sensor nodes move on a grid representing the streets of a city. In this case the size of a block is 200 m by 200 m. We define a junction as the intersection of a horizontal and a vertical street. When a target or sensor node reaches a junction, it randomly chooses one of the four directions (i.e., forward, backward, left, or right) with equal probability. The speed is chosen in the same way as for the Random Way-point model. We assume that buildings or other types of physical obstacles are located inside every block so that mobile sensors on one street cannot communicate with the mobile sensors on the other street. The mobile sensors on the junction area can communicate with the mobile sensors of both streets. We assume that the junction area is an open space 30 $m \times 30$ m square. We assume that the nodes that are within 15 m distance from a junction point are considered to be inside a junction area. Mobile sensors are initially randomly placed according to a uniform distribution on the plane in the case of scenarios based on the Constant Velocity and Random Way-point models or on points of the street grid in the scenarios based on the Manhattan model.

We evaluate the prediction error with different mobility models in order to study the impact of the choice of the underlying model for the Kalman filter predictor. The choice of the model of the predictor is fundamental and should be based on the knowledge of the movement patterns of the target. In what follows, we evaluate the impact of discrepancies

between the chosen model and the actual mobility patterns of the nodes.

Sections 4.5.1, 4.5.2, and 4.5.3 presents the evaluation of the correctness of the prediction of the target location. In these sections, we consider 144 sensors in the simulation area. We selected this density since it describes a scenario where from time to time the target is not in reach (i.e., a scenario characterized by medium "sparseness"). In Sections 4.5.4, 4.5.5, and 4.5.6, we analyze the performance of tasking and recovery focusing on the tracking duration and associated cost.

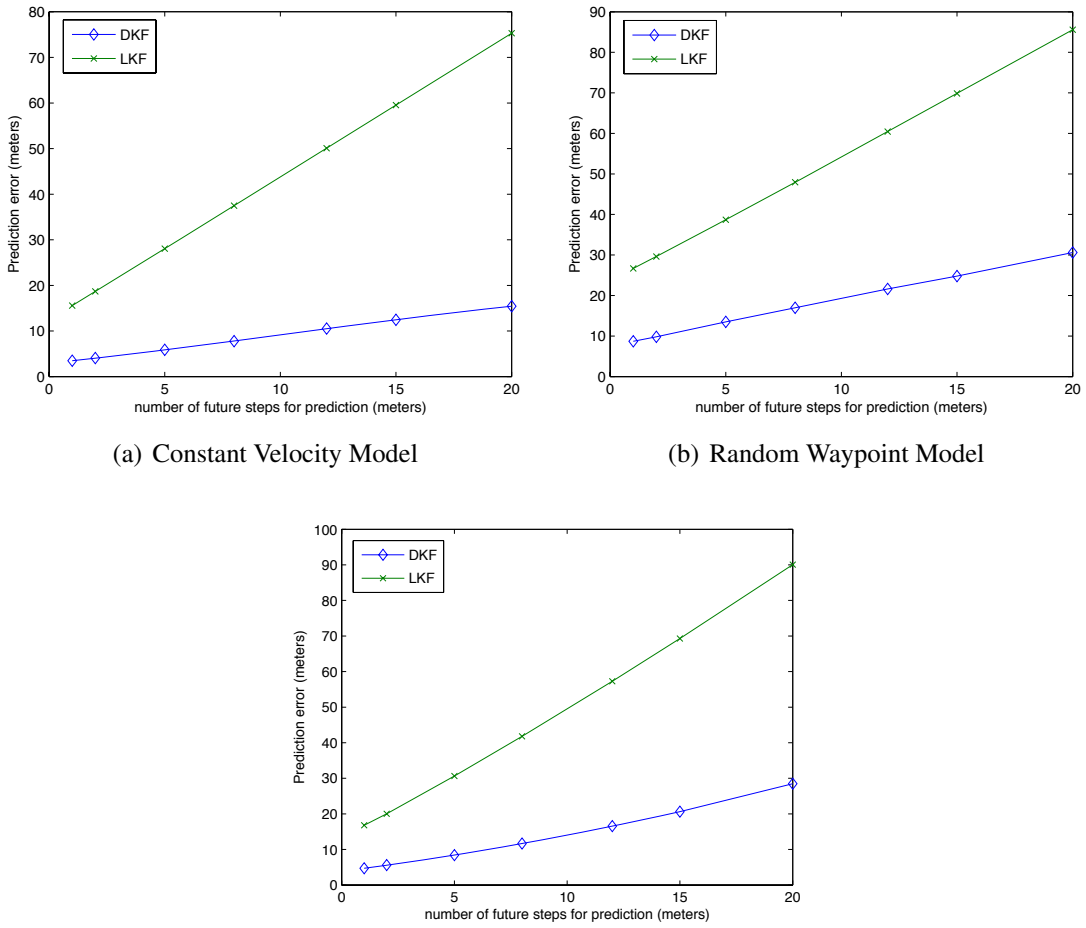


(a) Constant Velocity Model

(b) Random Waypoint Model

(c) Manhattan Model

Figure 4.10: Correlograms of the residual time series for the evaluation of the location predictability.



(a) Constant Velocity Model

(b) Random Waypoint Model

(c) Manhattan Model

Figure 4.11: Prediction Error of Target Location.

4.5.1 Accuracy of the Target Location Prediction

We examine the predictability of the positions of the target using the Local and Distributed Kalman filters by means of the predictability test method proposed in [71]. We firstly evaluate if the time series of the prediction error is a random process. This ensures that the prediction model is appropriate for the process under observation [20]. We test the predictability of the Local and Distributed Kalman filters when the target is actually moving according to Constant Velocity model used for the prediction model. Then, we also evaluate it using other models such as the Random Way-point model and the Manhattan model. For

all the three mobility models, the target and the mobile sensors move following the same model.

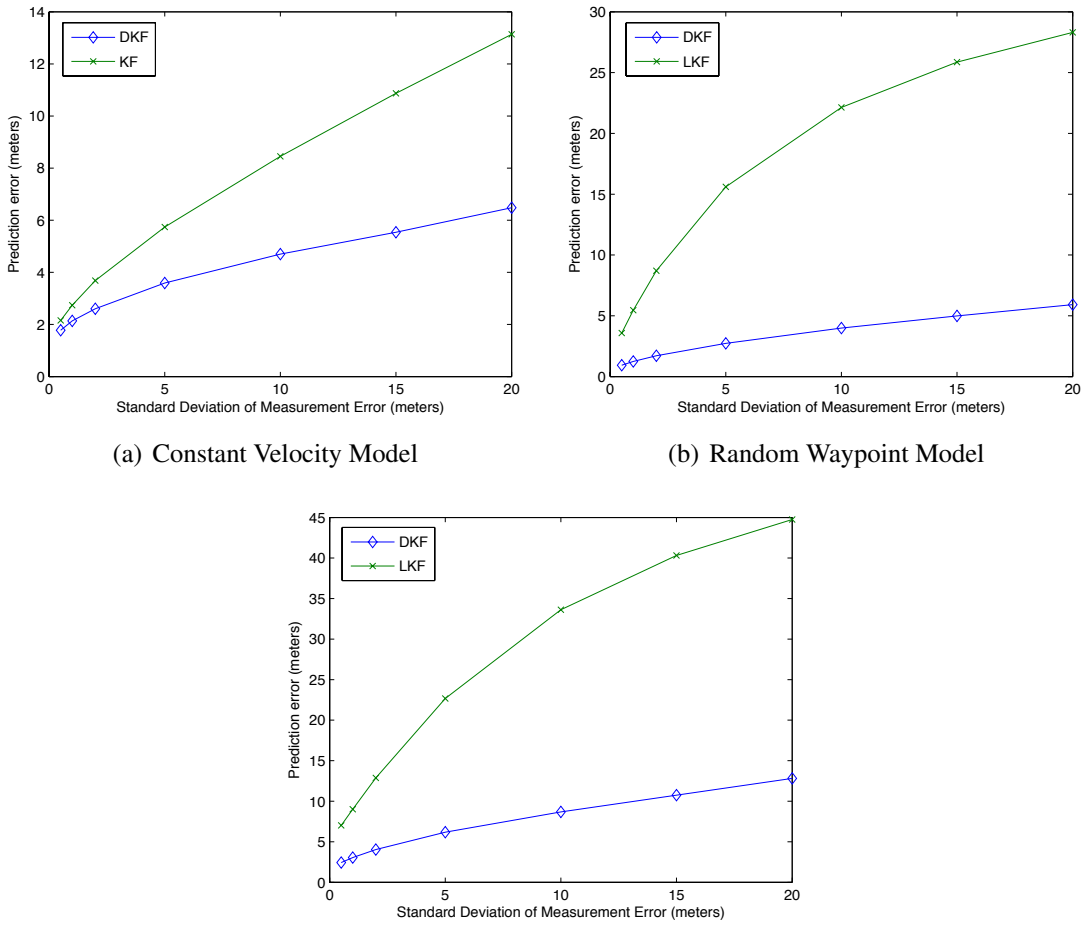
We select sensors that have tracked the target longer than 25 *s* consistently (i.e., they have a sufficient history of observations of the movements in order to be able to perform a valid analysis from a statistical point of view); for each of them, we calculate the autocorrelation coefficients. Figure 4.10 shows that the autocorrelation coefficients with a temporal lag from 1 *s* to 20 *s* of the time series of the prediction error based on the Local and Distributed Kalman filters are within the range of [-2/ $\sqrt{N}$, 2/ $\sqrt{N}$] (with $N = 25$ samples), which means that the residual time series (i.e., the time series of the prediction errors) is a random process. Therefore, with a sufficient number of samples, the movements of the hosts are predictable given the model that we are using in all the three scenarios.

We then analyze the accuracy of the prediction of the future target location considering the absolute values of the prediction errors. Figure 4.11 illustrates the prediction error of the Local and Distributed Kalman filters; as discussed above, both are based on the Constant Velocity model. We test their accuracy when the target is actually moving according to the Constant Velocity model; then, we test the prediction accuracy using other models such as the Random Way-point and the Manhattan models. The prediction error is measured in terms of the distance between the predicted target location and the actual target location. We average the prediction error calculated from each of the sensors that are tracking the target. The x-axis of the plot in Figure 4.11 is the number of future prediction steps (i.e, given an estimation at instant t, the prediction at instant $t + 1$, $t + 2$, etc.). The plot shows the prediction error as a function of the prediction step from 1 step to 20 steps ahead in the future. Distributed Kalman filter shows smaller prediction errors for the three scenarios. The prediction error of the the Local Kalman filter is more than 2.5 times greater than the Distributed Kalman filter in all cases. Both the Distributed and Local Kalman filters have slightly smaller prediction error for the Constant Velocity model than the other models as expected.

The accuracy of the prediction of the future location of the target relies on the accuracy of the estimation of the current target state (i.e., the location and the velocity) and the error in the prediction of the target location is due to the limited target detection time of each sensor. It is a rare occurrence that a sensor tracks a target from the beginning to the end or for a long period of time. Rather, each sensor participates in the tracking for a short period of time and the aggregated trace from these sensors can be used to reconstruct the whole trajectory of the target. For the Local Kalman filter, the sensors that are far away from the target cannot know the state of the target. As a result, when a newly tasked sensor detects the target for the first time and begins to track it, the initial error using the Local Kalman filter is potentially high and then it takes quite a long time for convergence of the filter and an error reduction. In contrast, when using the Distributed Kalman filter, sensors share the estimation of the state of the target with neighboring sensors and try to reach a consensus. The estimates of the newly tasked sensors using the Distributed Kalman filter converge quickly than using the Local Kalman filter. Therefore, the Distributed Kalman filter is able to reduce the prediction error, as shown in Figure 4.11.

4.5.2 Impact of the Measurement Error Variance

MetroTrack exploits mobile phones to track the target event. We use the mobile phone's microphone for acoustic sensing but these sensors are not designed as high quality sensors and, therefore, are characterized by significant measurement errors. As we experienced from measurements conducted using the N80 and N95 phones, the standard deviation of the measurement error (σ_R) can be large. We simulate MetroTrack with various measurement error variances and analyze their impact. Figure 4.12 shows the impact of the measurement error on target location prediction. The plot shows that the error of the prediction of the target increases for the Local Kalman filter as the measurement error variance increases while the error for the Distributed Kalman filter is not affected as much and remains low for



(a) Constant Velocity Model

(b) Random Waypoint Model

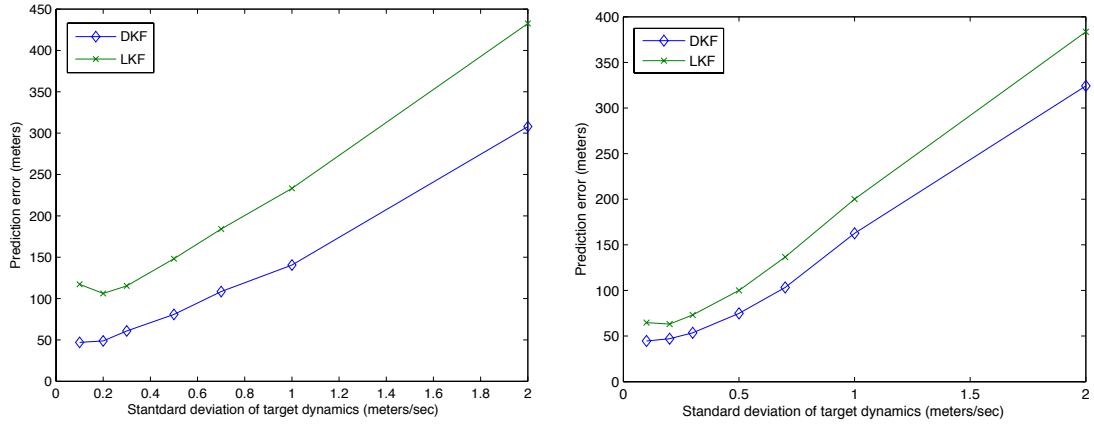
(c) Manhattan Model

Figure 4.12: Impact of Measurement Error (σ_R) on Target Location Prediction.

all the error variances. As we discussed in Section 4.3.2, sensors share the estimation of the state of the target with neighboring sensors and combine the prediction using the Distributed Kalman filter. The impact of prediction error outliers is reduced considerably. For the same reason, the Distributed Kalman filter is not affected by the measurement error variance as much as the Local Kalman filter.

4.5.3 Impact of the Target Dynamics

The impact of target dynamics (i.e., the variance of target movement) on target location prediction is analyzed in this section. Clearly, if the target is changing its movement



(a) Prediction Error (σ_R = 10) (b) Prediction Error (σ_R = 1)

Figure 4.13: Impact of Target Dynamics (σ_0).

directions and velocity more rapidly, the prediction error increases. We run simulations with Constant Velocity model using values from 0.1 to 2 for the standard deviation of target dynamics (σ_0) and measured the prediction error. The results in Figure 4.13 indicate that the prediction error of both filters are increased as the value of σ_0 increases and the increment is the same for both filters. It is interesting to note that the difference between the error of Distributed and the Local Kalman filters is not significant if the standard deviation of the measurement error (σ_R) is small (1 m/s), the difference is distinctively large if σ_R is large (10 m/s) (see Figure 4.13).

4.5.4 Tracking Duration

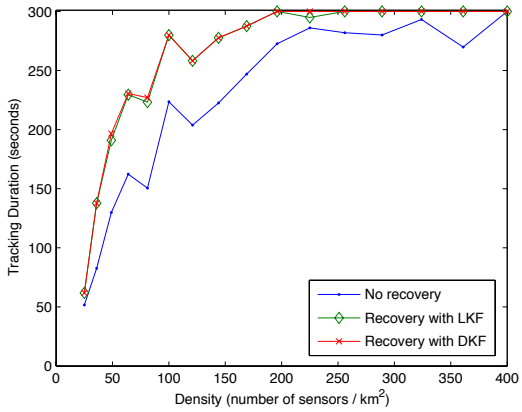
One of the main objectives of MetroTrack is to track the target for as long as possible without losing it. Therefore, the duration of tracking is one of the main performance metrics. Figure 4.14 shows the tracking duration with varying density and sensing range of mobile sensors with the Random Way-point model. We measure the duration of tracking when MetroTrack performs the information-driven tasking but it does not perform the prediction-based recovery (no recovery). We then measure the duration of tracking when MetroTrack performs the prediction-based recovery as well. We compare the tracking duration when

MetroTrack uses the Distributed Kalman filter (recovery with the Distributed Kalman filter) and when it uses the Local Kalman filter (recovery with the Local Kalman filter).

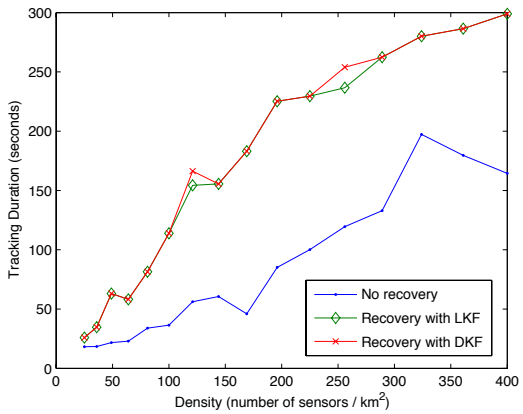
The results show how the tracking duration can be extended using the prediction-based recovery mechanism. The x-axis is the density of sensors and the y-axis is the duration of tracking. We run the simulation for 300 *s* and the tracking starts from the beginning of the simulation; the target is lost before the simulation ends. As we can see in these plots, the prediction-based recovery extends the duration of the tracking. Moreover, the recovery enables the tracking to last until the end of simulation with the density of greater than 200 sensors or more per km^2 if the sensing range is 100 *m* as shown in Figure 4.14(a). If the sensing range is 50 *m* as in Figure 4.14(b), the recovery enables the tracking to last until the end with the density of 400 sensors. The extended duration by the recovery process is longer for 50 *m* sensing range than for 100 *m* sensing range.

It is interesting to note that the recovery processes using both filters do not show much difference in tracking duration while the Distributed Kalman filter showed better accuracy in prediction. The forwarding zone is the sum of radius of recovery region, sensing range, and communication range as we explained earlier. This margin of error and the fact that the hosts at 1-hop are in recovery state reduce the impact of the inaccuracy of the prediction of the Local Kalman filter.

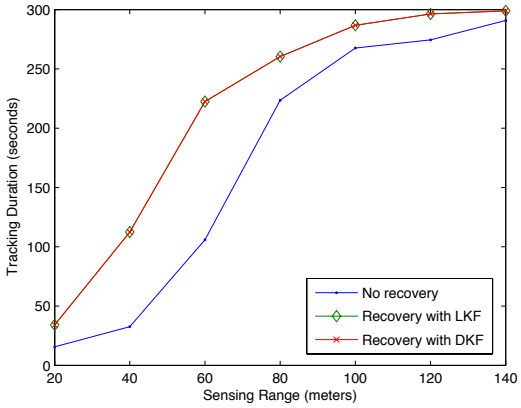
Figure 4.14(c) shows the tracking duration over varying sensing range with density of 144 nodes. The results show that the sensing range has a significant impact on the duration of the tracking and the recovery process becomes more important as the sensing range becomes smaller. If the sensing range is too small (such as 20 *m* in Figure 4.14(c)) for a given density, the recovery process is not always able to recover the target so the improvement is small.



(a) Sensing range of 100 *m*.



(b) Sensing range of 50 *m*.



(c) Density of 144 hosts.

Figure 4.14: Tracking duration vs. density and sensing range of mobile sensors.

4.5.5 Analysis of the Influence of the Recovery Area Radius

We discussed in Section 4.2.4 how MetroTrack sets the size of the recovery area radius. Figure 4.15 shows the impact of the choice of the recovery area radius. The density is 144 nodes in the simulated area. The duration is maximized if the radius is greater than 240 *m*. If we set the radius to be greater than 240 *m*, more sensors will receive and forward the recovery message. However, if we set the value of the radius greater than 240 *m*, more sensors would join the recovery process and the associated overhead (i.e., the energy used for sensing and broadcasting recovery messages) becomes greater. There is a trade-off between the overhead and the tracking duration. We argue that the tracking duration is a

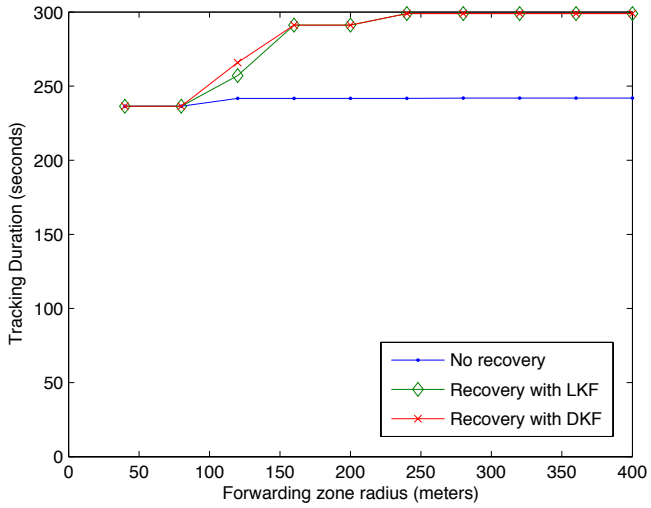


Figure 4.15: Tracking duration vs. recovery area radius.

more important performance metric as long as we do not task all the sensors in the area for the tracking of one target. We analyze such optimal radius for varying density (from 25 to 400 sensors in the simulation area) and the result was around 200 to 240 *m* for all the densities. In the simulation, we observe that the size of projected area is as small as 20 *m* at the beginning of the recovery phase and the size increases as time advances. We note that the radius that we observe in the testbed experimental is very close to the optimal radius that we derived from the simulation study.

4.5.6 Energy Cost

As discussed in Section 4.4, energy should not be considered a critical aspect of the design of this class of system, because the tracking process takes place occasionally and only when the target is in the vicinity of the phones. However, in this section we present an evaluation of the energy cost associated to the tracking process that can be valuable for deployment scenarios where the frequency of targets to be tracked is high. We simulated MetroTrack to analyze this energy cost considering scenarios characterized by different densities. We only show the performance results of MetroTrack using the Distributed Kalman filter because

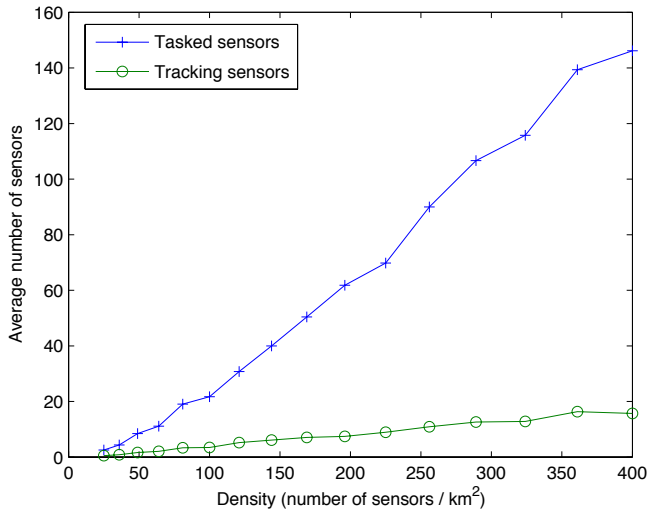


Figure 4.16: Number of active sensors vs. density.

they are almost identical to those related to the Local Kalman filter.

Figure 4.16 shows the number of sensors that are tasked and the number of sensors that are actually detecting the target. Among the nodes in the simulated area, only one third of the sensors are tasked. It is important to note that MetroTrack tasks the mobile sensors only when the target is currently being detected or have been detected recently (before the recovery timer expires). Hence, all the mobile sensors will be idle most of the time when the target event is not present in the area. The tracking sensors (i.e., the sensors that are currently detecting the target) broadcast the task messages and the state of the target for the Distributed Kalman filter calculation every sampling period (which is 1 s in the simulation). Figure 4.16 shows that this overhead is fairly small because only 5 % (or less) of the sensors in the simulation area are tracking the target.

As we mentioned in Section 4.2.4, the sensors may be in recovery process although the target is already recovered. In addition, the sensors may initiate the recovery process even if the target is not lost. MetroTrack uses the suppression process to stop those sensors from continuing the recovery process by broadcasting the recovery messages. We analyze how many recovery messages are sent if MetroTrack does not perform the suppression process

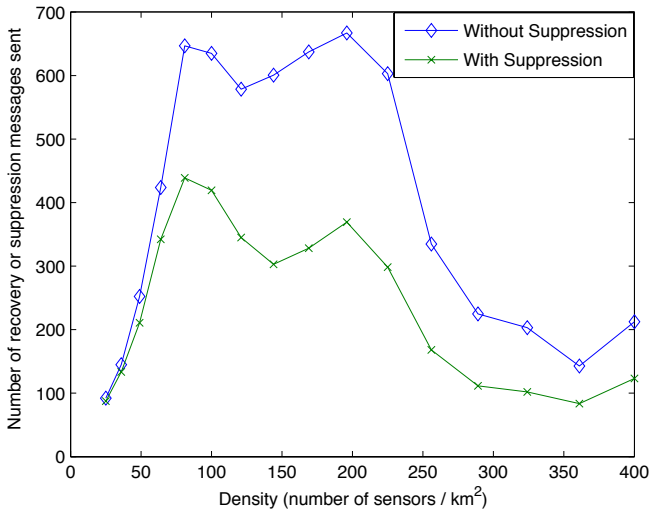


Figure 4.17: Overhead of recovery process vs. density.

and how much the suppression process can reduce the number of recovery messages. Figure 4.17 shows the number of recovery messages and suppression messages sent in scenarios characterized by different density. The y-axis in the figure is the total number of messages sent during the simulation, which lasts for 300 *s*. The results indicate that the number of the recovery or suppression messages sent is reduced by half with most of the density scenarios, which means that the suppression process is preventing the sensors from staying in recovery process when it is not necessary.

4.6 Related Work

Tracking mobile events using static sensor fields or using mobile sensors with controlled mobility (e.g., sensors on automated vehicles and airplanes) have been reported widely in the literature. However, to the best of our knowledge there has been little or no research on tracking mobile events using uncontrolled mobility, in particular using mobile devices carried by people such as mobile phones and PDAs.

In [19], the Frisbee model addresses the energy constraint problem of tracking mobile events in static sensor networks. Frisbee is a conceptual zone that moves along with the

target based on a disc-shaped area. In the Frisbee model, the majority of sensors sleep and tracking is initiated by sentry nodes that remain awake. The sentry nodes inside the Frisbee area send a wake-up signal to the sensors that fall inside the Frisbee-shaped area (hence the name). MetroTrack follows the Frisbee model in the sense that the wake-up signal can be viewed as the simplest form of tasking messages. The MetroTrack tasking messages contain event descriptors supporting a richer semantics with respect to the Frisbee model.

VigilNet [43] is a tracking system that realizes the Frisbee model using wake-up signal and sentry sensors. In [65], a distributed group management algorithm is proposed for scalable collaboration of sensors. The geographical group formed by this algorithm can be considered as another instantiation of the Frisbee model. In [109], the communication tree is optimized and managed within a geographical group. All three proposals involve the election and the maintenance of a leader for the group. In MetroTrack, the sensors that are tasked for the same target can be viewed as a group but the group itself does not require a leader since all the computation is performed locally and is fully distributed. We think the reliance on group leaders and election complicates the scalability and robustness of the solution space. For that reason, MetroTrack is guided by peers as equals, local processing, and distributed processing.

The existing work on tracking mobile events using controlled mobility is focused on controlling the movement of the mobile sensors (e.g., mobile robots) to get the best resolution in terms of tracking. In [108], a gradient-driven tracking algorithm is proposed to enable a single mobile sensor to move closer to the source of the event. In [111], a distributed mobility management scheme to improve the quality of tracking is proposed. Each node makes a decision on their next move that can improve the quality of tracking considering the risks of losing connectivity with neighbors as well as losing sensing coverage of certain regions of the sensor field.

MetroTrack also addresses the problem of disruptive tracking due to uncontrolled mobility that can create temporary holes in the emergent topology of the mobile sensor

network. This problem has been studied in static sensor networks, even if the results have not directly been applied to (mobile) event tracking. Corke et al. proposed an algorithm to find holes in static sensor networks [26]. Mobicast [47] proposes a mechanism to disseminate information to the nodes around a connectivity hole in static networks. In mobile networks, the shape of the hole may change dynamically, so using the static knowledge of the topology metric may not be feasible. MetroTrack focuses on recovering the lost target based on prediction of the movement of the target rather than using *a priori* knowledge of the topology.

4.7 Conclusion

In this chapter, we proposed MetroTrack, the first distributed tracking system that tracks mobile events using off-the-shelf mobile phones. We presented the design and implementation of the system, discussing the mathematical foundations that our distributed prediction models are based on. We evaluated the system through the deployment of a prototype implementation of the system using Nokia N80 and N95 mobile phones and analyzed the performance of the system for a number of different scenarios through simulation. While the MetroTrack prototype implementation focused on tracking a mobile audio source as a proof of concept we believe that the algorithms and techniques discussed in this chapter are more broadly applicable to an emerging class of problems related to the efficient tracking mobile events using off-the-shelf mobile devices such as mobile phones, PDAs, and mobile embedded sensors.

Chapter 5

Conclusion

This thesis has presented three information-driven systems in sensor networks and mobile ad hoc networks, namely, MetroTrack, Funneling-MAC, and SWAN. Although the three systems address different challenges in different contexts, they share the fundamental design principle that the systems utilize information from a dynamic environment, which may affect the performance and robustness of the systems.

In MetroTrack, each mobile sensor makes a local decision about whether to forward the task using local state information and sensor reading in order to respond to dynamically moving targets and time-varying device densities around the target. In the SWAN model, the rate control mechanism uses local MAC delays as feedback information and ECN-based regulation uses local aggregated traffic rate to decide whether to trigger the regulation procedure. In the funneling-MAC, the sink node collects traffic pattern and intensity information inside the funneling region and uses the information to dynamically compute the schedule and new depth of the intensity region. Therefore, both SWAN and funneling-MAC are capable of dynamically responding to the time-varying traffic conditions.

Chapter 2 has presented SWAN, a stateless wireless ad hoc network model that provides service differentiation supporting real-time services over best-effort services. Real-time services have certain bandwidth and delay requirements. However, a conventional network

model that uses TCP-like congestion control ensures maximum system throughput but at the cost of larger packet delays. SWAN uses distributed-control algorithms to ensure that the delays are close to the minimum, while the system throughput approaches the maximum.

Unlike stateful, reservation-based QoS support approaches, SWAN is designed to handle both real-time UDP traffic and best-effort UDP and TCP traffic without the need for the introduction and management of per-flow state information in the network. Therefore, SWAN does not involve complex signaling or state-controlling mechanisms that stateful approaches require. To the best of our knowledge, SWAN is the first stateless service differentiation system in mobile ad hoc networks. SWAN supports real-time services without the support of a QoS-capable MAC to deliver service differentiation. SWAN assumes a best-effort MAC and performs feedback-based control mechanisms to support real-time services with bandwidth and delay requirements.

The SWAN model consists of three distributed algorithms, i.e., (1) feedback-based local rate control; (2) sender-based admission control; and (3) dynamic ECN-based regulation. The local rate control is based on the well-known additive increase and multiplicative decrease (AIMD) rate control mechanism. In order to satisfy the bandwidth and delay requirements of real-time UDP traffic, rate control of TCP and UDP best-effort traffic is performed locally on every mobile node in a fully-distributed and decentralized manner. SWAN rate control uses per-hop MAC delay as a feedback for local rate control mechanism. Rate control is designed to force the traffic rate of best-effort traffic below a threshold rate such that the necessary bandwidth is set aside to support real-time traffic. Rate control also allows the best-effort traffic to efficiently utilize the rest of the bandwidth that is not utilized by the real-time traffic at any particular moment. SWAN performs sender-based admission control for real-time UDP traffic. The sender of a real-time session sends a probe packet to the destination. The probe packet travels back to the sender with information about the traffic condition along the path of the real-time session. The sender-based rate control makes the admission decision of the real-time session using the information that the probe

packet collected. SWAN does not have excessive signaling overhead because the sender-based admission control does not require that the intermediate node maintain per-flow state. Instead, SWAN performs ECN-based regulation to dynamically regulate admitted real-time sessions in the face of network dynamics caused by mobility or traffic-overload conditions. When the aggregated bandwidth of admitted real-time traffic exceeds the threshold rate, the ECN mechanism forces the senders to re-establish or drop the real-time session of the sender.

The performance evaluation of SWAN using NS-2 simulator and a wireless testbed has been presented in Chapter 2. In addition, the analytical assessment of the MAC delay and busy probability has been presented, confirming the effectiveness of the SWAN model. The simulation, analytical, and experimental results show that real-time traffic experiences low and stable delays under various traffic and mobility conditions with SWAN.

Chapter 3 has addressed the problem of funneling effect in sensor networks. Typical multi-hop wireless sensor networks exhibit a unique funneling effect that is the result of the distinctive many-to-one, hop-by-hop traffic pattern found in these networks. The funneling effect results in a significant increase in transit traffic intensity, collisions, congestion, packet loss, and energy drain as events move closer toward the sink. While network (e.g., congestion control) and application techniques (e.g., aggregation) can help counter this problem, they cannot fully alleviate it. We took a different but complementary approach to solve this problem and presented the design, implementation, and evaluation of a localized, sink-oriented, funneling-MAC capable of mitigating the funneling effect and boosting application fidelity in sensor networks.

The funneling MAC is based on a CSMA/CA operating network-wide, with a localized TDMA algorithm overlaid in the funneling region, which is the area within a small number of hops from the sink. In this sense, the funneling-MAC represents a hybrid-MAC approach, but it does not have the scalability problems associated with the network-wide deployment of TDMA. The funneling-MAC is “sink-oriented” because the burden of managing the

TDMA scheduling of sensor events in the funneling region falls on the sink node, not on the resource-limited sensor nodes. The funneling-MAC is also "localized" because TDMA only operates locally in the funneling region close to the sink, not across the complete sensor field. Another interesting contribution of the funneling-MAC is that its meta-schedule advertisement algorithm addresses the boundary issue in the area where two heterogeneous MAC protocols coexist in the sensor networks. A hybrid TDMA/CSMA runs in the funneling region and seamlessly coexists with pure CSMA outside of that region. The potential interference caused by dynamically increasing or decreasing the funneling region is effectively managed using meta-schedule advertisement.

Experimental results using various topologies and traffic rates are presented in Chapter 3. The results indicate that the funneling-MAC is capable of mitigating the funneling effect, improving throughput, reducing losses, and enhancing energy efficiency. Importantly, the funneling-MAC significantly outperforms other MAC protocols, such as B-MAC and Z-MAC.

Chapter 4 presented MetroTrack, the first sensor-based, mobile-event tracking system using off-the-shelf mobile phones. Event tracking is an important application and active research topic in sensor networks. Conventional tracking applications have driven the deployment of sensor networks across a number of disparate domains, such as battlefields, industrial facilities, and protected environmental areas. Yet, event tracking in urban areas is at an early stage of development that presents a new set of challenges. When we think about the tracking problem, traditional solutions that come to mind are based on the deployment of static sensor networks. However, static networks used in urban areas have significant cost, scaling, coverage, and performance issues that will limit their deployment. MetroTrack presents the potential of an alternative approach that leverages sensor-enabled mobile phones as mobile sensors to track mobile events in urban areas to overcome the limitations of the static sensor network approach.

MetroTrack addresses a number of challenges associated with building a mobile-event

tracking system using mobile phones. First, mobile sensors must be tasked before sensing can begin, and, for the system to scale effectively, only those mobile sensors near the target event should be tasked. Another challenge is that there is no guarantee that there will be sufficient density of mobile sensors around any given event of interest because the mobility of mobile sensors is uncontrolled. MetroTrack is capable of efficiently tasking only the mobile sensors in close proximity to a target event of interest by means of an information-driven tasking. MetroTrack is predicated on the fact that a target will be lost during the tracking process, and thus it takes compensatory action to recover the target, allowing the tracking process to continue.

MetroTrack is based on two fully-distributed algorithms, i.e., (1) information-driven tasking and (2) prediction-based recovery. The tasking procedure is information-driven because each sensor node, according to its local sensor reading state, independently determines whether to forward the tracking task to its neighbors or not. Therefore, the solution is fully-distributed and uses local state only. The recovery procedure is based on a prediction estimated by a distributed Kalman-Consensus filtering algorithm that estimates the lost target and its margin of error.

We analyzed the algorithm's performance by means of large-scale simulations using different mobility models and through the deployment of a prototype testbed using Nokia N80 and N95 phones. The prototype testbed demonstrates the proof-of-concept that the tracking of mobile events in an urban area using off-the-shelf mobile phone is feasible. Simulation results indicate that MetroTrack is robust in the presence of different mobility models and device densities.

In this dissertation, we address three important problems in wireless ad hoc and sensor networks. Wireless ad hoc and sensor networks are expected to provide unprecedented capability to perceive and share the information around our daily lives as well as from the physical environment at fine granularity and large scale. SWAN, Funneling-MAC, and MetroTrack offer important building blocks in realizing the capability of wireless ad hoc

and sensor networks.

Chapter 6

My publications as a Ph.D candidate

My publications as a Ph.D. candidate are listed below. The list also includes research papers that are indirectly related to the work presented in this thesis such as the papers related to INSIGNIA and MetroSense project.

6.1 Journal Papers

- Gahng-Seop Ahn, Andrew T. Campbell, Andras Veres and Li-Hsiang Sun. Supporting Service Differentiation for Real-Time and Best Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN), *IEEE Transactions on Mobile Computing (TMC)*, Special Issue of Best Wireless Papers from INFOCOM 2002, Vol. 1, No. 3, pp. 192-207, July-September 2002.

- Gahng-Seop Ahn, Emiliano Miluzzo, Andrew T. Campbell, Se Gi Hong, and Francesca Cuomo. A Localized, Sink-Oriented MAC for Mitigating the Funneling Effect in Sensor Networks, *ACM Transactions on Sensor Networks (TOSN)* (under submission).

- S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, Gahng-Seop Ahn, and A. T. Campbell. BikeNet: A Mobile Sensing System for Cyclist Experience Mapping,

ACM Transactions on Sensor Networks (TOSN) (under submission).

- Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang, and Andrew T. Campbell. INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks, *Journal of Parallel and Distributed Computing (Academic Press)*, Special issue on Wireless and Mobile Computing and Communications, Vol. 60 No. 4 pg. 374-406, April 2000.

6.2 Magazine papers

- Seoung-Bum Lee, Gahng-Seop Ahn, and Andrew T. Campbell. Improving UDP and TCP Performance in Mobile Ad Hoc Networks with INSIGNIA, *IEEE Communication Magazine*, June 2001.

- Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristof Fodor, Gahng-Seop Ahn. People Power - The Rise of People-Centric Sensing, *IEEE Internet Computing*, Special Issue on Mesh Networks, July/August 2008.

6.3 Conference and Workshop Papers

- Gahng-Seop Ahn, Andrew T. Campbell, Andras Veres and Li-Hsiang Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks, *IEEE Infocom 2002*, New York, New York, June 2002.

- Gahng-Seop Ahn, Emiliano Miluzzo, Andrew T. Campbell, Se Gi Hong and Francesca Cuomo. Funneling-MAC: A Localized, Sink-Oriented MAC for Boosting Fidelity in Sensor Networks, *Fourth ACM Conference on Embedded Networked Sensor Systems (Sensys 2006)*, Boulder, Colorado, November 2006.

- Gahng-Seop Ahn, Andrew Campbell, Hong Lu, Mirco Musolesi, and Reza Olfati-Saber. Predictive Tracking of Mobile Events using Mobile Phones, *Sixth ACM Conference on Embedded Networked Sensor Systems (Sensys 2008)*, (under submission).

- Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang, and Andrew T. Campbell. Evaluation of the INSIGNIA Signaling System, *8th IFIP International Conference on High Performance Networking (Networking 2000)*, Paris, France, May 2000.

- S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, Gahng-Seop Ahn, and A. T. Campbell. The BikeNet Mobile Sensing System for Cyclist Experience Mapping, *Fifth ACM Conference on Embedded Networked Sensor Systems (SenSys 2007)*, Sydney, Australia, November 2007.

- Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald A. Peterson, Gahng-Seop Ahn, and Andrew T. Campbell. MetroSense Project: People-Centric Sensing at Scale, *ACM SenSys Workshop on World-Sensor-Web (WSW 2006)*, Boulder, Colorado, November 2006.

6.4 IETF Internet Draft

- Gahng-Seop Ahn, Andrew T. Campbell, Andras Veres and Li-Hsiang Sun. SWAN, Internet Draft, *draft-ahn-swan-manet-00.txt*, February 2003.

- Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang and Andrew T. Campbell. INSIGNIA, Internet Draft, *draft-ietf-manet-insignia-01.txt*, IETF MANET Working Group Document, November 1999.

References

[1] Tarek Abdelzaher, Yaw Anokwa, Pter Boda, Jeff Burke, Deborah Estrin, Leonidas Guibas, Aman Kansal, Samuel Madden, and Jim Reich. Mobiscopes for Human Spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[2] G.-S. Ahn, A.T. Campbell, A. Veres, and L.-H. Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks. Internet Draft, draft-ahn-swan-0.1.txt, work-in-progress, September 2002.

[3] G.-S. Ahn, A.T. Campbell, A. Veres, and L.-H. Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks. In *Proc. of IEEE INFOCOM 02*, 2002.

[4] Gahng-Seop Ahn, Emiliano Miluzzo, Andrew T. Campbell, Se Gi Hong, and Francesca Cuomo. Funneling-MAC: A Localized, Sink-Oriented MAC for Boosting Fidelity in Sensor Networks. In *Proc. of 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, 2006.

[5] K. A. Arisha, M. A. Youssef, and M. F. Younis. Energy-Aware TDMA-Based MAC for Sensor Networks. In *Proc. of IEEE Workshop on Integrated Management of Power Aware Communications, Computing and NeTworking (IMPACCT 2002)*, 2000.

[6] F. Bai, N. Sadagopan, and A. Helmy. IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks. In *Proceedings of INFOCOM 2003*, San Francisco, CA, USA, April 2003.

[7] D. Bansal and H. Balakrishnan. TCP-Friendly Congestion Control for Real-Time Streaming Applications. Technical Report MIT-LCS-TR-806, MIT Laboratory for Computer Science, May 2000.

[8] D. A. Beyer. Accomplishments of the DARPA SURAN program. In *Proc. of the Military Communications Conference (MILCOM)*, September 1990.

[9] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[10] G. Bianchi, A. Capone, and C. Petrioli. Throughput Analysis of End-to-End Measurement-Based Admission Control in IP. In *Proc. of IEEE INFOCOM 00*, 2000.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP) Version 1 Functional Specification. RFC 2205, September 1997.

[12] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint Admission Control : Architectural Issues and Performance. In *Proc. of ACM SIGCOMM 00*, August 2000.

[13] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multihop Wireless Ad Hoc Network Routing Protocols. In *Proc. of ACM/IEEE Intl Conf. Mobile Computing and Networking (MobiCom)*, October 1998.

[14] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: a Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *Proc. of 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, 2006.

[15] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing (WCMC). Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[16] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, Ronald Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristof Fodor, and Gahng-Seop Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing Special Issue on Mesh Networks*, July/August 2008.

[17] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, and Ronald A. Peterson. People-centric urban sensing. In *Proceedings of WICON'06*, page 18, New York, NY, USA, 2006. ACM.

[18] C. Centinkaya and E. Knightly. Egress Admission Control. In *Proc. of IEEE INFOCOM 00*, 2000.

[19] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat Monitoring: Application Driver for Wireless Communications Technology. In *Proceedings of ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[20] Chris Chatfield. *The Analysis of Time Series An Introduction*. Chapman and Hall, 2004.

[21] S. Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE Journal on Special Areas in Communications*, 17(8), August 1999.

[22] X. Chen. Dual near field effect in radio frequency simulations. In *2002 Summer Computer Simulation Conference*, 2002.

[23] CHIPCON. CC1000 Data Sheet: CC1000 Single Chip Very Low Power RF transceiver. http://focus.ti.com/lit/ds/swrs048a/swrs048a.pdf.

[24] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks*, 1989.

[25] Chee-Yee Chong and Srikanta P. Kumar. Sensor Networks: Evolution, Opportunities, and Challenges. 91(8):1247–1256, August 2003.

[26] P. Corke, R. Peterson, and D. Rus. Finding holes in Sensor Networks. In *Proceedings of the Workshop on Omniscient Space: Robot Control Architecture Geared toward Adapting to Dynamic Environments at ICRA 2007*, April 2007.

[27] Dana Cuff, Mark Hansen, and Jerry Kang. Urban Sensing: Out of the Woods. *Communications of the ACM*, 51(3):24–33, 2008.

[28] T. Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, 2003.

[29] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proc. SIGCOMM Symp. Comm. Architectures and Protocols*, September 1989.

[30] C. T. Ee and R. Bajcsy. Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. In *Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[31] S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. MetroSense Project: People-Centric Sensing at Scale. In *Proceedings of Workshop on World-Sensor-Web (WSW 2006)*, October 2006.

[32] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G-S. Ahn, and A. T. Campbell. The BikeNet mobile sensing system for cyclist experience mapping. In *Proceedings of SenSys '07*, pages 87–101, New York, NY, USA, 2007. ACM.

[33] Shane E. Eisenman and Andrew T. Campbell. E-CSMA: Supporting Enhanced CSMA Performance in Experimental Sensor Networks using Per-neighbor Transmission Probability Thresholds. In *Proc. of IEEE INFOCOM 2007*.

[34] A. El-Hoiydi and J.D. Decotignie. WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. In *In Proc. of the Ninth International Symposium on Computers and Communications (ISCC 2004)*, 2004.

[35] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In *Proc. of 5th Symposiumon Operating Systems Design and Implementation (OSDI 2002)*, 2002.

[36] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Comm. Rev.*, 24(5), October 1994.

[37] Christian Frank, Philipp Bolliger, Christof Roduner, and Wolfgang Kellerer. Objects calling home: Locating objects using mobile phones. In *Proceedings of the 5th International Conference on Pervasive Computing (Pervasive 2007)*, LNCS. Springer, Toronto, Canada, May 2007.

[38] S. Gandham, M. Dawande, and R. Prakash. Link Scheduling in Sensor Networks: Distributed Edge Coloring Revisited. In *Proc. of IEEE INFOCOM 2005*, 2005.

[39] M. Gerla, X. Hong, and G. Pei. Landmark routing for large ad hoc wireless networks. In *Proc. of the IEEE GLOBECOM 2000*, November 2000.

[40] Lewis Girod, Martin Lukac, Vlad Trifa, and Deborah Estrin. The design and implementation of a self-calibrating distributed acoustic sensing platform. In *Proceedings of the 4th international conference on embedded networked sensor systems (SenSys'06)*, pages 71–84, New York, NY, USA, 2006. ACM.

[41] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, (2):388–404, 2000.

[42] G. Halkes and K. Langendoen. Crankshaft: An Energy-Efficient MAC-Protocol For Dense Wireless Sensor Networks. In *In Proc. of 4th European Conference on Wireless Sensor Networks (EWSN 07)*, January 2007.

[43] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh. VigilNet: An Integrated Sensor Network System for Energy-Efficent Surveillance. *ACM Transactions on Sensor Networks*, 2004.

[44] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. Abdelzaher. Achieving Real-Time Target Tracking Using Wireless Sensor Networks. *ACM Transaction on Embedded Computing System (TECS)*, 2007.

[45] Tian He, J.A. Stankovic, Chenyang Lu, and T. Abdelzaher. Landmark routing for large ad hoc wireless networks. In *Proc. of the 23rd International Conference on Distributed Computing Systems*, May 2003.

[46] B. Hohlt, L. Doherty, and E. Brewer. Flexible Power Scheduling for Sensor Networks. In *Proc. of IEEE Information Processing in Sensor Networks (IPSN '04)*, 2004.

[47] Q. Huang, C. Lu, and G-C Roman. Spatiotemporal Multicast in Sensor Network. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[48] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[49] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons INC., 1991.

[50] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, pages 153–181, 1996.

[51] John Jubin and Janet D. Tornow. The DARPA packet radio network protocols. 75(1):21–32, January 1987.

[52] K. Jamieson and B. Hull and H. Balakrishnan. Understanding the real-world performance of carrier sense. In *ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND 2005)*.

[53] R Bagrodia M Gerla M Bereschinsky K Xu, K Tang. Adaptive bandwidth management and QoS provisioning in large scale ad hoc networks. In *In proc. of Military Communications Conference (MilCom)*, October 2003.

[54] Rudolph. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, March 1960.

[55] S. Kalyanaraman, D. Harrison, S. Arora, K. Wanglee, , and G. Guarriello. A One-Bit Feedback Enhanced Differentiated Services Architecture. Internet Draft, draft-shivkuma-ecn-diffserv-01.txt, work-in-progress, March 1998.

[56] Aman Kansal, Arun A. Somasundara, David D. Jea, Mani B. Srivastava, and Deborah Estrin. Intelligent fluid infrastructure for embedded networks. In *Proceedings of MobiSys'04*, pages 111–124, New York, NY, USA, 2004. ACM.

[57] R.H. Katz. Adaptation and Mobility in Wireless Information Systems. 1(1):6–17, 1994.

[58] F. Kelly, P. Key, and S. Zachary. Distributed Admission Control. *IEEE Journal on Selected Areas in Communications*, 18(12), December 2000.

[59] Y.-B. Ko and N.H. Vaidya. Geocasting in Mobile Ad Hoc Networks: Location-based Multicast Algorithms. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA'99)*, February 1999.

[60] Sunil Kumar, Vineet S. Raghavan, and Jing Deng. Medium Access Control protocols for ad hoc wireless networks: A survey. 4(3):326–358, May 2006.

[61] Seoung-Bum Lee, Gahng-Seop Ahn, X. Zhang, and Andrew T. Campbell. INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks. *Journal of Parallel and Distributed Computing, special issue on wireless and mobile computing and communications*, 60(4):374–406, April 2000.

[62] J. Li, C. Blake, D. D. Couto, H. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2001)*, 2001.

[63] J. Li and G. Lazarou. A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks. In *Proc. of IEEE Information Processing in Sensor Networks (IPSN '04)*, 2004.

[64] C.R. Lin and M. Gerla. A Distributed Architecture for Multimedia in a Multihop Dynamic Packet Radio Network. In *Proc. of IEEE Globecom 95*, pages 1468–1472, November 1995],.

[65] Juan J. Liu, Jie Liu, J. Reich, P. Cheung, and F. Zhao. Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications. In *Proceedings of IPSN'03*, April 2003.

[66] G. Lu, B. Krishnamachari, and C. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks. In *Proc. of 4th International IEEE Workshop on Agorithms for Wireless, Mobile, Ad-Hoc and Sensor Networks (WMAN 2004)*, 2004.

[67] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Ieee standard 802.11, June 1999.

[68] Medium Access Control (MAC) Enhancements for Quality of Service (QoS). Draft supplement to ieee standard 802.11, November 2001.

[69] Emiliano Miluzzo, Xiao Zheng, Kristof Fodor, and Andrew T. Campbell. Radio Characterization of 802.15.4 and its Impact on the Design of Mobile Sensor Networks. In *Proceedings of Fifth European Conference on Wireless Sensor Networks (EWSN 2008)*, Bologna, Italy, January 2008.

[70] Global mobile information systems program. http://www.darpa.mil/ito/research/glomo/ index.html.

[71] Mirco Musolesi and Cecilia Mascolo. Evaluating Context Information Predictability for Autonomic Communication. In *Proceedings of 2nd IEEE Workshop on Autonomic Communications and Computing (ACC'06)*, Niagara Falls, NY, June 2006. IEEE Computer Society Press.

[72] S. Nanda, D.J. Goodman, and U. Timer. Performance of PRMA: A Packet Voice Protocol for Cellular Systems. *IEEE Transactions on Vehicular Technology*, 40, August 1991.

[73] Glenn Nofsinger and George Cybenko. Distributed chemical plume process detection. In *Proceedings of MILCOM'05*, pages 1076–1082, 2005.

[74] Nokia. Python for S60. http://wiki.opensource.nokia.com/projects/PyS60.

[75] R. Olfati-Saber. Distributed Kalman Filtering for Sensor Networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, December 2007.

[76] R. Olfati-Saber. Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility. In *Proceedings of the 2007 American Control Conference*, July 2007.

[77] G. Pei and C. Chien. Low Power TDMA in large wireless sensor networks. In *Proc. of IEEE Communications for Network Centric Operation MILCOM 2001*, 2001.

[78] C.E. Perkins and E.M. Royer. Ad-Hoc On Demand Distance Vector Routing. In *Proc. of IEEE Workshop Mobile Computing Systems and Applications*, February 1999.

[79] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey. Data funnelling: Routing with Aggregation and Compression for Wireless Sensor Networks. In *Proc. of IEEE Sensor Network Protocols and Applications (SNPA 2003)*, 2003.

[80] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[81] Purdue University. Cell phone sensors detect radiation to thwart nuclear terrorism, January 2008. http://news.uns.purdue.edu/x/2008a/080122FischbachNuclear.html.

[82] V. Rajendran, J. J. Garcia-Luna-Aceves, and K. Obraczka. Energy-Efficient, Application-Aware Medium Access for Sensor Networks. In *Proc. of the 2nd IEEE Conf. on Mobile Ad-Hoc and Sensor Systems (MASS 2005)*, 2005.

[83] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks. In *Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, 2003.

[84] K. Ramakrishnan, S. Floyd, and D. Black. An Addition of Explicit Congestion Notification (ECN) to IP. Internet Draft, draft-ietftsvwg-ecn-03.txt, work-in-progress, March 2001.

[85] R. Ramanathan and M. Streenstrup. Hierarchically-Organized, Multi-Hop Mobile Wireless Networksfor Quality-of-Service Support, 2002.

[86] S. Ramanathan. (T/F/C)DMA channel assignment in wireless networks. In *In proc. of IEEE INFOCOM 1997*, April 1997.

[87] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-MAC: a Hybrid MAC for Wireless Sensor Networks. In *Proc. of 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, 2005.

[88] I. Rhee, A. Warrier, and L. Xu. DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks. In *In proc. of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2006.

[89] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, RFC 3261, June 2002.

[90] Samarth H. Shah, Kai Chen, and Klara Nahrstedt. Dynamic Bandwidth Management in Single-Hop Ad Hoc Wireless Networks. 10(1-2):199–217, 2005.

[91] N. Shrivastava, C. Buragohain, D. Agrawat, and S. Suri. Medians and Beyond: new aggregation techniques for sensor networks. In *Proc. of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[92] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A Reliable-Transport Protocol for Wireless Wide-Area Networks. In *Proc. of ACM ACM/IEEE Intl Conf. Mobile Computing and Networking (MobiCom)*, August 1999.

[93] J.L. Sobrinho and A.S. Krishnakumar. Quality-of-Service in Ad Hoc Carrier Sense Multiple Access Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1353–1368, August 1999.

[94] Y. Sun, O. Gurewitz, and D.B. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *In Proc. of 6th ACM Conference on Embedded Networked Sensor Systems (SenSys 2008)*, November 2008.

[95] Andras Veres, Andrew T. Campbell, Michael Barry, and L.-H. Sun. Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control. *IEEE Journal on Selected Areas in Communications, special issue on mobility and resource management in next-generation wireless systems*, 19(10):2094–2104, October 2001.

[96] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft. Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks. In *Proc. of 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, 2005.

[97] Funneling-MAC webpage. http://www.cs.dartmouth.edu/ sensorlab/funneling-mac/.

[98] SWAN Project webpage. http://comet.columbia.edu/swan.

[99] Z-MAC webpage. http://www4.ncsu.edu/ rhee/export/zmac/index.html.

[100] TinyOS website. http://tinyos.net.

[101] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2001)*, 2001.

[102] A. Woo and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, 2003.

[103] Y.-J. Kim and R. Govindan and B. Karp and S. Shenker. Geographic routing made practical. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI 05)*, Berkeley, CA, USA, 2005.

[104] Yaling Yang and R. Kravets. Distributed QoS guarantees for realtime traffic in ad hoc networks. In *In proc. of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, October 2004.

[105] Yaling Yang and R. Kravets. Contention-aware admission control for ad hoc networks. 4(4):363–377, 2005.

[106] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of IEEE INFOCOM*, 2002.

[107] W. Ye, F. Silva, and J. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *Proc. of 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, 2006.

[108] Q. Zhang, G. Sobelman, and T. He. Gradient-Driven Target Acquisition in Mobile Wireless Sensor Networks. In *Second International Conference on Mobile Ad Hoc and Sensor Networks (MSN 2006)*, December 2006.

[109] W. Zhang and G. Cao. DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks. *IEEE Transactions on Wireless Communication*, September 2004.

[110] M. Zhao and SD Servetto. An analysis of the maximum likelihood estimator for localization problems. *2nd International Conference on Broadband Networks*, pages 59–67, 2005.

[111] Y. Zou and K. Chakrabarty. Distributed Mobility Management for Target Tracking in Mobile Sensor Networks. *IEEE Transactions on Mobile Computing*, 6(8):872–887, 2007.