ROUTING AND SCHEDULING FOR REAL-TIME DATA DISSEMINATION IN SENSOR NETWORKS

BY

KE LIU

B.S. in Computer Science, Fudan University, Shanghai, China 2000

THESIS

Submitted in partial fulfillment of the requirements for

the degree of Master of Science in Computer Science

in the Graduate School of

Binghamton University

State University of New York

2005

Accepted in partial fullfillment of the requirements for
the degree of Master of Science in Comptuter Science
in the Graduate School of
Binghamton University
State University of New York
2005

Dr. Nael Abu-Ghazaleh _____

Date _____

Department of Computer Science.

Dr. Kyoung-Don Kang _____

Date _____

Department of Computer Science.

Abstract

Wireless sensor networks have been attracting significant research and commercial interest recently. In many sensor network applications, such as real-time surveillance and monitoring, the sensed data need to be delivered with some real-time constraints, such as an end-to-end deadlines. Since the data dissemination model is different from the trandi-tional wireless ad hoc networks or any centralized systems, new challenges arise. Different solutions have been proposed, such as real-time sensitive routing protocols, data packet prioritization, and real-time scheduling.

This thesis demonstrates that real-time scheduling in sensor networks is influenced by several factors: (1) the routing protocol can influence the paths taken in the network and the interaction between contending flows; and (2) the packet scheduling algorithm can arbitrate the use of the network in a deadline sensitive manner to ensure that most packets make their deadline. Most existing work in real time scheduling has focused on the second factor, with little work addressing the first factor and the interaction of the two. Further, to prioritize and schedule the data traffic so as to meet the end-to-end deadline, the end-to-end delay should be tracked carefully. As observed, the end-to-end delay of a data packet travelling from the source to destination is influenced by several factors: (1) the physical and MAC layer transmission delay; (2) the processing delay at each hop; and (3) the queueing delay at each hop. While most existing work attempts to minimize the end-to-end delay, it often ignores the queueing delay as a major component of the overall delay.

In this thesis, we first study the effect of routing protocols on real-time sensor network

applications. We compare stateful and stateless routing protocols, showing the advantage and disadvantage of both. Two routing protocols that are representative of these classes, shortest path routing and greedy forwarding routing, are studied real-time constraints. We then study existing scheduling algorithms for sensor networks and analyzes the components of the end-to-end delay of a data packet from the source to destination. We conclude that existing scheduling algorithms do not consider all those possible contributing parts and therefore do not lead to effective solutions under moderate to high contention. Finally, the thesis proposes a new scheduling algorithm called Just-in-Time Scheduling (JiTS), which deals with all the contributing components of the end-to-end travelling delay of data packets in sensor network. It compares JiTS with another prioritizing and scheduling algorithm for real-time data dissemination in sensor networks, velocity monotonic scheduling. It simulates both scheduling algorithms based on two typical routing protocols, shortest path routing and greedy forwarding with NS2.

For my parents, DESHU LIU *and* MIN YU

without whom none of this would have mattered.

Acknowledgement

It is my great pleasure to thank all the people who helped me actualize in this thesis. I am deeply indebted to my advisor, Dr. Nael Abu-Ghazaleh, for his guidance and support extended to me during the course of my research work. His creative suggestions along with his words of encouragement, patience and time went a long way in writing this thesis. Thanks Nael. I consider it my good fortune to have got an opportunity to work with him.

The idea of this thesis is first intuited by the course CS580t: Real-time system. I thank Dr. Kyoung-Don Kang who gave a creative suggestion in the study of real-time methods for sensor networks.

I would thank my labmate Sameer Tilak, who is an expert in the research of sensor networks. He spent a lot of time disscussing my ideas, gave me advice in coding with the NS2 simulator, leading to my trenmendous progress of this thesis. I also thank Vinay Kolar and Kaushik Balasubramanian, my labmates and good friends who gave me much help.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

Wireless sensor networks are an important emerging technology that will revolutionize sensing for a wide range of scientific, military, industrial and civilian applications[1]. A large number of inexpensive sensors collaborating on sensing a phenomena provide cost-effect detailed monitoring of the area under observation [2, 3]. While some sensor networks are deployed to collect information for later analysis [4], most applications require monitoring or tracking of phenomena in real-time [5].

Improvements in hardware technology have led to the development of low-cost, miniature micro-sensor nodes which are composed of a single tiny chip (typically several mm^2) with embedded memory (typically $10KBytes \sim$ several $100KBytes$), microprocessor, sensing transducer, and a transceiver[6]. The typical power supply is based on battery on chip, generating several 100 milliwatts. Collaboration and coordination of sensors in sensor networks is essential to energy-efficient operation [7]. Coordinating hundreds or even thousands of these sensor nodes is a difficult challenge.

In this chapter, we first introduce some research challenges in sensor networks with respect to real-time dissemination. We then describe overview the proposed approach and summarize our primary findings.

1.1 Challenges in Sensor Networks

Challenges to, and effective solutions for, building large-scale distributed system such as the Internet are well understood. However, those solutions do not translate well to sensor network environments due to the severely constrained resources in terms of CPU, memory, storage, communication bandwidth and available energy. The system design challenges are presented differently from those posed by the exiting computer networks. The design of sensor networks has the following features [7]:

- *Data-Centric:* the identity (such as IP address) of the individual nodes in a traditional computer networks are necessary because of the end-to-end nature of these networks. In sensor networks, the sensors are merely an instrument for monitoring the physical phenomena: an observer is interested in the **data** generated by sensors, and not in the individual sensors. In fact, ideally, the application is not even aware of the sensors, posing its queries in terms of application-specific parameters (e.g., what is the temperature in this region?).

- *Application-Specific:* Unlike traditional networks which are designed generally for a wide variety of applications, a sensor network is typically designed for a special set of applications sharing the same behaviors. Because of the resource limitations, this application specific nature invites specialized "vertical" solutions that take advantage of the application characteristics to optimize operation.

In general, the following design limitations are faced in sensor network system design [8, 9]:

- *Resources Constraints:* As the thesis already mentioned, the resources in short include power supply, computation ability, memory/storage, communication consumption. The cost of the system may be also a challenge due to the possible large number of nodes deploying in the network. Even the improvement in hardware manufacture

is great recently, these constraints are still tight. Solutions have to make tradeoff among these resources constraints. Since the nature of *application-specific*, the solution would be different for variant applications.

- *Limited access:* For some sensor network applications, the sensor nodes would be deployed into a hostile area. To recharge or recollect sensor nodes is impossible. Access to any individual sensor node is very hard so that system needs to be designed carefully. Energy consumption may be the core of the design of these systems. Security is also an important concerning.

- *Unreliability and Network dynamics:* Sensor nodes are unreliable due to the limited resources. The structure of network may change dynamically for sensors that are moving (e.g., tied to animals), dying or appearing (e.g., after a recharge period, or as new nodes are deployed). The challenge is how to collaborate and self-organize to continue to provide acceptable performance in the face of these dynamics.

- *Scale and variable density:* The density of nodes may vary across systems, and even within a single system. This may occur, for example, because of deployment of sensors from a plane or a moving vehicle. The solutions must be able to adapt to such situations.

- *Real-time:* Many applications require the sensor network to respond within real-time constraints. Due to the limited storage at the sensor nodes, real-time data traffic may be the main traffic in the sensor network. The solutions need to disseminate the real-time data traffic efficiently. The primary challenges here are how to prioritize and schedule packets. Moreover, due to the nature of the shared wireless medium, routing essentially provides the chance to spatially schedule the packets to reduce contention for the network resources.

1.2 Limitations of Existing Solutions

A primary challenge in real-time sensor network applications is how to carry out sensor data dissemination given source-to-sink end-to-end deadlines when the communication resources are scarce. Although routing/data transport solutions have been proposed in the context of wireless ad hoc networks, the characteristics of sensor networks make the problem different. The traffic patterns in sensor networks in response to queries or events are different from the point-to-point communication typical of sensor networks. Moreover, the bursty nature of traffic in sensor networks, as the degree of observed activity varies, can cause the network resources to be exceeded. In addition, the ad hoc nature of multi-hop sensor networks makes it difficult to schedule network traffic centrally as in traditional real-time applications.

One of the proposed solutions for real-time data dissemination [10] prioritizes packet transmission at the MAC layer according to the deadline and distance from the sink. This work have several limitations: (1) While packets are prioritized, they are not delayed – when traffic is bursty, high contention results, increasing transmission and queuing delays. Furthermore, packets generated by different sensors at the same time (e.g., in response to a detected event), can lead to high collision rates. Jittering such packets can help reduce this hot-spotting; (2) MAC level solutions cannot account for the queuing delay in the routing layer (which occurs above the MAC layer); these delays can have a significant impact on end-to-end delay especially under high load; and (3) MAC level solutions require re-engineering of the sensor radio hardware and firmware, making deployment difficult and potentially causing interoperability problems with earlier hardware that supports different MAC protocols.

Since the scheduling needs to consider the queueing delay in the routing layer which is above the MAC layer, the impact of the routing protocols used must be carefully examined. The effect of the routing protocol on the real-time scheduling success is not suf-

ficiently understood. Some existing solutions [11][12][13] for routing in real-time traffic context provide non-deterministic routing as an extension of stateless geographic-based routing protocols. More specifically, these approaches use the best next hop with respect to the traffic/congestion situations, not only the geographic proximity as per the greedy Geographical Forwarding protocol. In addition, Geographical Forwarding, which is used in these solutions, does not always lead to the shortest delay paths, making it more difficult to meet the deadline. Furthermore, when using a longer path in terms of number of hops, increased contention for the medium results as more transmissions are needed to forward a packet.

1.3 Proposed Approach and Thesis Contributions

The primary contribution of this thesis is a new Just-in-Time Scheduling (JiTS) approach for real-time data dissemination in sensor networks that addresses many of the shortcomings of the existing solutions. JiTS delays packets at intermediate hops (not just prioritizes them) for a duration that is a function of their deadline. Delaying packets allows the network to avoid hotspotting while maintaining deadline-faithfulness. A second contribution of the thesis is to explore the role of the routing protocol in the success of real-time scheduling in sensor networks: we discover that it plays a critical role that in some instances outweighs the effect of the scheduling algorithm. We demonstrate the performance of JiTS in comparison to state of the art in this area (the RAP and SPEED models [10, 11]) using simulation.

JiTS delays packets at every hop for a duration of time which is a function of the number of hops to the sink and the deadline. JiTS uses an estimate of the MAC layer transmission delay and accounts for it when deciding how long to delay a packet. By delaying the packets, rather than prioritizing them, JiTS achieves the following advantages: (1) A full estimate of the delay is used, including the delay due to queuing in the network

layer; (2) The load is distributed over the available deadline time, potentially allowing the network to tolerate transient periods of high contention gracefully, and to avoid transient hot-spotting; (3) It provides packets with a longer chance to wait for correlated packets for purposes of aggregation or packet combining; and (4) It works well with different types of routing protocols, stateful or stateless, providing better performance by using shortest path (hop-count) routing, without any change of the existing popular MAC layer or lower layers protocols.

The base JiTS algorithm distributes the slack time (available time before the deadline expires) uniformly across all the hops. However, in a gathering data collection pattern, the amount of contention is typically higher the closer to the sink. Therefore, a second contribution of this thesis is to explore policies that allocate the time non-uniformly among the hops, to provide more time to the hops closer to the sink. As the data packets approaching the sink, more queue delay time is given which leads to tolerate more possible contentions of resources.

The performance of JiTS is evaluated against RAP[10] with different deadline constraints and routing protocols. The third contribution of the thesis is to show that Shortest Path (hop-count) routing outperforms Geographical forwarding Routing with respect to real-time traffic. Since any other geographic-based routing protocols would route data packets through longer pathes to resolve the path failure (void problem) in Geographical forwarding routing, using a path failure free deployment can typically show the generality. JiTS outperforms RAP both in terms of deadline miss ratio, and overall packet goodput. Further, the nonlinear delay version of JiTS is able to achieve the best performance. These results hold across different network topologies (regular and random), and different traffic conditions. In particular, when the traffic generation is bursty, JiTS performance far out-paces that of RAP.

JiTS is developed and evaluated with some (typical) assumptions about the routing

protocol and the sensor network. In the following, we discuss the implications of these assumptions and the possibility of extending JiTS to operate under different conditions. JiTS requires an estimate of the distance to the destination, preferrably in terms of hop count, from the routing protocol. However, alternative forms of distance indicator, such as geographical distance, can be used to allow JiTS to interoperate with geographically based routing protocols. For the scenarios where the end-to-end deadline cannot be obtained through the data packets itself, or is not explicit, JiTS may not make accurate decisions; however, the lack of this information would similarly impact other real-time solutions as well. We did not explore the effect of multi-path routing on the performance of JiTS. In addition, for some applications, sensor nodes are aggressively duty cycle controlled to provide energy saving. While we do not pursue solutions in such an environment we believe that JiTS can be extended to operate successfully in them. The JiTS scheduler keeps some data packets in the queue, which would force the sensor nodes to be awake when it is time transmit them. JiTS can be adapted to schedule packets only when the node is scheduled to be awake. Moreover, the sleep time can be incorporated in the delay estimate. Finally, JiTS can feedback the activity information to the duty cycle controller to improve or reduce the duty cycle dependingon the level activity.

1.4 Overview of the Remainder of the Thesis

The broad outline of the thesis is as follows: Chapter 2 provides the background of the wireless sensor networks. Chapter 3 describes some routing protocols designed for ad hoc wireless networks including sensor networks. Chapter 4 discusses some important real-time scheduling algorithms, for both the centralized and distributed system. Chapter 5 describes the proposed JiTS architecture. Chapter 6 compares the performance of JiTS to a famous real-time scheduling architecture RAP, and shows the simulation results of SPEED. Finally Chapter 7 concludes.

Chapter 2 provides the background of the wireless sensor networks. First the thesis introduces a popular real-world hardware platform, UCB motes, and its software platform, TinyOS. Then it describes briefly the Media Access Control protocol used for sensor networks, IEEE 802.11. After that, the thesis studies some localization algorithms which is a main service required by most sensor network applications.

Chapter 3 describes some routing protocols designed for ad hoc wireless networks including sensor networks. First, one source-driven routing protocol, Dynamic source routing(DSR)[14], is introduced as a typical routing protocol working in a wireless ad hoc network. Then a simple but typical routing protocol for sensor networks, shortest path routing (SP) is described . Since the former two routing protocols are both stateful routing protocols, which need the intermediate nodes to maintain some end-to-end states of pathes, a set of stateless routing protocols, geometric routing protocols [15–20], are studied. Since the research of sensor networks now is a hot pot, the thesis introduces some new routing protocols proposed recently[21, 22]. Finally in this chapter, the thesis studies two routing protocol designs for real-time application in sensor network: SPEED[11, 12] and MMSPEED [13].

Chapter 4 first describes the Rate Monotonic Scheduling, a typical scheduling algorithm for a centralized system. Then it discusses the differences between the centralized system scheduling to distributed ones, especially for the data communication model in the wireless ad hoc networks such as sensor networks. A traditional scheduling algorithm for the data disseminating in the wireless networks, deadline-based scheduling is introduced briefly. Then a very importance scheduling algorithm designed for sensor networks, Velocity Monotonic Scheduling, in studied carefully. The thesis analyzes the different roles of transmission delay and queueing delay in real-time data dissemination for different scenarios.

Chapter 5 introduces the details of the proposed JiTS architecture. The experimental

study evaluating the role of the routing protocols and evaluating JiTS in comparison to RAP and SPEED is presented in Chapter 6. Finally, conclusions and future work are described in Chapter 7.

Chapter 2

Background

Sensor networks differ from traditional wireless ad hoc networks [23] because of their data-centric nature and their limited resources [24]. In addition, the unique data dissemination models that result from their role as monitors of real-world phenomena (e.g., data gathering model and the event based dissemination model [25]) leads to different data networking protocols used for routing and data-dissemination.

In this chapter, first we introduces a real-world hardware/software platform of sensor networks: Berkely motes. The reason for describing this complete system is to provide an appreciation of a state of the art sensor system platform (hardware, system software and development software). We then focus on two aspects of operation that are relevant to the real-time scheduling problem: (1) We present a typical MAC layer protocol used in sensor networks. MAC operation can significantly influence packet delay; and (2) We describe a basic service needed by sensor networks applications: *localization*, or the ability of sensors to determine their location. In addition to being needed by the application to interpret the sensor data, localization is needed for geographical routing protocols. Since we study the effect of routing protocols on real-time scheduling, understanding the effect of localization on geographical routing protocols is necessary to understand their expected behavior.

Two aspects of operation that are most relevant to real-time scheduling in sensor net-

works: (1) routing; and (2) packet scheduling are not discussed in this Chapter. Instead, they are treated separately and in detail in the following two chapters.

2.1 Hardware resources available for tiny sensors

The hardware resources available on micro-sensor nodes are significantly more limited than other wireless devices such as PDAs or laptops. This limitation is driven by cost and size considerations. For example, consider an application such as object tracking in a hostile environment: the cost of the sensors must be acceptable to allow deployment of many sensors that will likely be unrecoverable. Moreover, the size of the sensors should be small so that they are not easily detected and neutralized. The computing ability of the sensor nodes are very limited, memory is often insufficient for complex operations. Moreover, the available storage may not allow large volumes of data to be buffered and data must be relayed to collection points for analysis and storage. The challenge is how to balance the application requirements against the limited available resources. Since transmission is one of the most costly operations (transmitting 1 byte 100 meters consumes similar energy as processing several thousand instructions []), it is often the primary target for reducing the consumed energy; there is an emphasis on collaboration to reduce the overall data size.

A sensor node typically consists of the following components [25, 26]:

- Transducer: an electronic device converting a physically measured phenomena into a measurable electrical quantity. It is used to provide samples of the observed phenomena. Common examples are microphones, thermometers, and accelerometers.

- Microprocessor: Each sensor node typically includes a general microprocessor to allow customized in-network processing (necessary for energy-efficient operation). The microprocessor defines the computation ability of a sensor node. Due to the cost of a single node, and other concerns such as energy efficiency, a small microprocessor with limited computational power is typically used.

- Memory/Storage: to process the sensory data locally, storage methods are needed at each sensor node. The amount of storage present on a single node affects the overall performance of network (e.g., the available buffer space). The amount of available memory and storage is typically small, up to a few Megabits. Nonvolatile storage is not available in many cases.

- Transceiver (radio transmitter/receiver): the locally processed sensory data must be relayed to interested observers, if they are needed in real-time, or because the local storage space is limited. The transceivers are used to communicate the data to other sensors and eventually to a basestation or an observer. The tansmission range influences the required network density to provide robust operation. It is typically up to a few hundred feet.

- Power supply: the power supply is also very limited because of weight, form factor and cost considerations. While there is research into energy harvesting, most existing sensor nodes have non-rechargable batteries. These define the lifetime of sensor nodes and eventually the whole system.

2.2 Media Access Control in wireless sensor networks

Modified versions of the IEEE 802.11 MAC layer protocol[27] are often used by sensor network platforms. This section describes the MAC and the physical (PHY) layer functionality of IEEE 802.11. While 802.11 supports both infrastructured and ad hoc modes, some features which are specific to infrastructured networks are not enabled in the Ad hoc mode. The 802.11 MAC layer specification specifies two kinds of access methodologies as follows.

1. Point Coordination Function(PCF): This is usually used for real time data transmission with priorities in infrastructured networks. This is a contention free access pro-

tocol; PCF is not used in Ad hoc networks and is therefore not discussed in detail.

2. Distributed Coordination Function(DCF): All Ad hoc networks (including sensor networks) use DCF as the access methodology. This is a contention based access protocol.

This section describes the features of DCF.

2.2.1 Carrier Sensing

In Ethernet, channel access is carried out using Carrier Sense Multiple Access with collision Detection (CSMA/CD). A node first senses the channel when it has a packet to transmit. If the channel is busy, then the transmission is delayed, otherwise the packet is transmitted. Collisions may still occur because of concurrent or nearly concurrent sensing of an idle channel. If a collision is detected, packet transmission is aborted and the packet is retransmitted after an exponential backoff. This algorithm works well in wired networks where the each node can hear every other node. In Ad hoc networks, this assumption does not hold true because the interference among nodes is defined by their physical location and the surrounding environment. For example, if node A is in range with nodes B and C, this does not mean that B and C are in range with each other. Thus, the channel may be idle at a transmitter but not at the receiver, and vice versa (this gives rise to the well-known hidden terminal and exposed terminal problems [28] discussed in more detail below)

As a result of these limitations, wireless networks use *Carrier Sense Multiple Access with Collision Avoidance*(CSMA/CA). The carrier sensing works in the same way as in wired internet. The node that wants to transmit a packet will listen to the channel. If the channel is idle for a certain duration of time called *Double Inter-Frame space*(DIFS), then the node assumes that no other node nearby is trying to transmit and goes ahead with its transmission. The recipient of the packet will acknowledge(ACK) the packet if the packet is received without any errors.

2.2.2 Virtual Carrier sensing

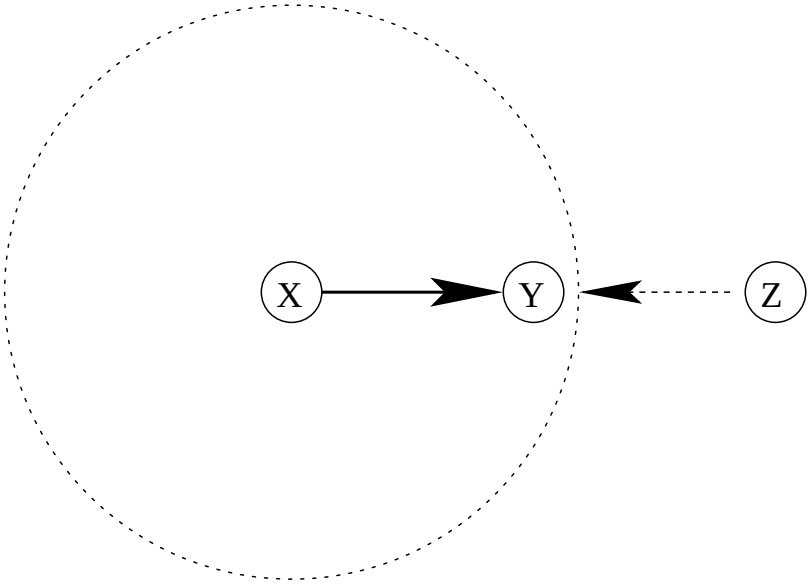


FIG. 2.1. Need for virtual carrier sensing

Consider the example in Figure 2.1 where node X and node Z wants to send a packet to node Y. Node X will start the transmission of the packet. Node Z is not within the range of X and therefore cannot listen to node X's transmission. If Z has a packet, it assumes that the channel is free and start transmission; this will cause a collision with the ongoing X-Y communication.This problem is termed as *Hidden Terminal Problem* since Z is *hidden* from node X-Y's communication. Hence, a basic ACK scheme with CSMA/CA will not work satisfactorily. To overcome this problem, a scheme called as "Collision Avoidance" is used. This scheme attempts to temporarily acquire the channel by letting the neighboring nodes of sender and receiver know about the ongoing transmission. It is accomplished by sending small control packets before sending the data.

When a sender wants to transmit a packet, it sends a control packet called **Request to Send (RTS)** to the receiver if the channel is sensed as free. This packet informs senders within transmission range of the sender of the expected transmission. Upon receiving the

RTS, the receiver will send a **Clear to Send(CTS)** packet back to the sender if the physical channel is not busy; this packet in turn informs potential interferers around the destination of the incoming packet, which helps address the hidden terminal problem. The sender then sends the **DATA** packet to the receiver which respons with an **Acknowledgment (ACK)** packet indicating correct reception of the data packet. This idea was first proposed in MACA protocol [29] and was later refined by MACAW protocol [30]. Before sending the RTS, the node will sense the channel for DIFS time period. For sending other packets, the node will sense for smaller time called *Short Inter Frame Space(SIFS)*. The shorter delay after RTS ensures that a packet that is successful in sending an RTS does not get preempted after that stage by another packet.

Each node maintains a table called as the *Network Allocation Vector (NAV)* which aids Virtual Carrier sensing. The RTS and CTS frames contain a *duration* field in their header which specifies the time interval needed to complete the complete RTS-CTS-DATA-ACK handshake. Any node that listens to these packets will update the NAV as described below. Upon listening to this RTS packet, the neighbors of the sender will know that one of the node near them is about to transmit data and also the duration of handshake. During this time, if the node sends some packet, then it may interfere with an ongoing communication. Hence it will mark the time in duration field in its NAV. This time is the period for which the node needs to be silent. When any node listens to CTS, it updates its NAV in the same manner.

2.2.3 Exponential backoff

Backoff is a standard solution for regulating traffic in contention based networks. In Ad hoc network, the node senses the medium before sending the RTS when it needs to send a new data packet. If the medium keeps idle for a DIFS period, the node begins its transmission at once. If the channel is sensed busy then the node will remain silent for

a random number of *slots* and will set its backoff timer accordingly. The *slots* is chosen from 0 to a maximum value called *Contention Window (CW)*. If the node listens to any other transmission (channel busy) during this backoff time, then the backoff timer is frozen and it will be restarted when the channel becomes idle again. If the node does not get any kind of acknowledgment (CTS for an RTS or ACK for a DATA), which means collision happened, then the backoff timer is exponentially increased. It is doubled every time such situation occurs until reaching a maximal value. The backoff timer is capped at a maximum value to avoid very high backoffs.

2.2.4 MAC layer – Retransmission

Every time there is a failure to send the data to the receiver, either because of collision or because of failure to get acknowledgment packets (CTS or ACK), the backoff timer is increased exponentially as described above. The MAC layer tries to retransmit the same data packet for a given number of times given by a constant value called *Retransmit limit*. If the number of retransmissions exceeds the Retransmit Limit, then the MAC layer gives up sending that data packet (a *Retransmit packet drops*) and reports the failure to the above routing layer. Such a packet drop is interpreted as a mobility related failure by some routing protocols and is called *No RouTE error or NRTE*. Receiving such information typically causes the routing layer to initiate route repair mechanisms.

2.3 Localization in sensor networks

In self-configuring sensor network applications, sensors need to establish their location in order to allow observers to interpret their data meaningfully. More relevant to us is that location information is necessary for geographical routing – a routing approach that appears particularly suited to sensor networks because it requires only local interactions between nodes in direct range with each other. Two very important research works, RAP [10]

and SPEED [11], involve the geographical routing which is definitely based on localization service. The thesis introduces some localization research work here.

Two straightforwards methods to obtain the location information are manual configuration and Global Positioning System (GPS) [31]. Manual configuration is easy and low-cost. But it is not scalable and is a poort fit for dynamic/self configurating networks: any change in the location of a node, requires the re-configuration. Using GPS device is a good choice for dynamic location service, but it comes at a signficant cost and only works in outdoor settings. The alternative employed in many sensor networks is to conduct some kind of localization algorithm to determine location. The most common localization algorithm alternatives include:

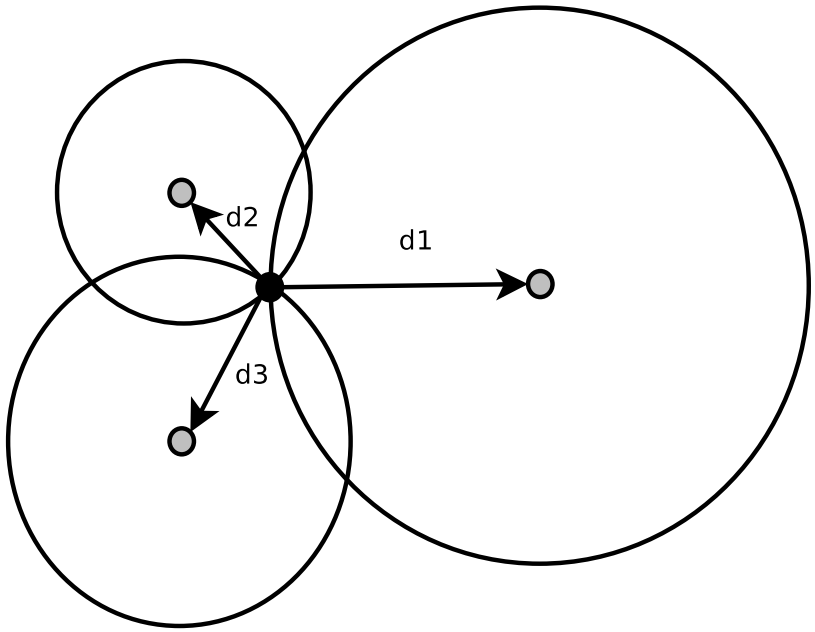


FIG. 2.2. Localization Algorithm: triangulation

- Triangulation: Location determined using trigonometry (lateration or angulation). In the network, some anchor nodes equipped some positioning device such as GPS have their accurate location information. These anchor nodes periodically broadcast

their location information to all other nodes in the network. Lateration[32, 33] is the calculation of position information based on distance estimated from the anchors. A 2D position requires three distance measurements, example shown in figure 2.2. Several approaches are used to estimate distance such as relative signal strength, and time difference of arrival. Angulation uses angle of arrival information and also requires angles from three known anchors in order to localize.

- Proximity Analysis: Instead of relying on sophisticated approaches to measure either distance or angle of arrival from anchors, this approach relies on tracking the mere presence of anchors only. By figuring out what beacons are nearby, heuristics can be employed to provide a coarse grained estimate of location (e.g., by computing the centroid of the received beacon locations [34]).

- Other Approaches: Approaches such as scene analysis and dead reckoning have also been employed. In scene analysis, observed features are used to infer location using a precomputed map. In the differential scene analysis algorithms, the differences between successive scenes are compared to each other to calculate location. This kind of localization algorithm requires a compiling database which is not available for most sensor network applications. In dead reckoning, the initial location of a mobile is known, and motion sensors (accelerometer) are used to track the velocity. The velocity is used to estimate distance using dead-reckoning to interpolate between velocity measurements.

A typical proximity localization algorithm used for sensor network is the Ad-Hoc Positioning System(APS)[31], or called *DV-hop*. It is a multihop localization schemes using a distance-vector flooding technique to determine the minimum hop count and average hop distance to known beacon positions. Each beacon broadcasts a packet with its location and a hop count, initialized to one. The hop-count is incremented by each node as the packet is

forwarded. Each node maintains a table of minimum hop-count distances to each beacon. A beacon can use the absolute location of another beacon along with the minimum hop count to that beacon to calculate the average distance per hop. The beacon broadcasts the average distance per hop, which is forwarded to all nodes. Individual nodes use the average distance per hop, along with the hop count to known beacons, to calculate their local position using lateration. An example of APS is shown in figure 2.3.

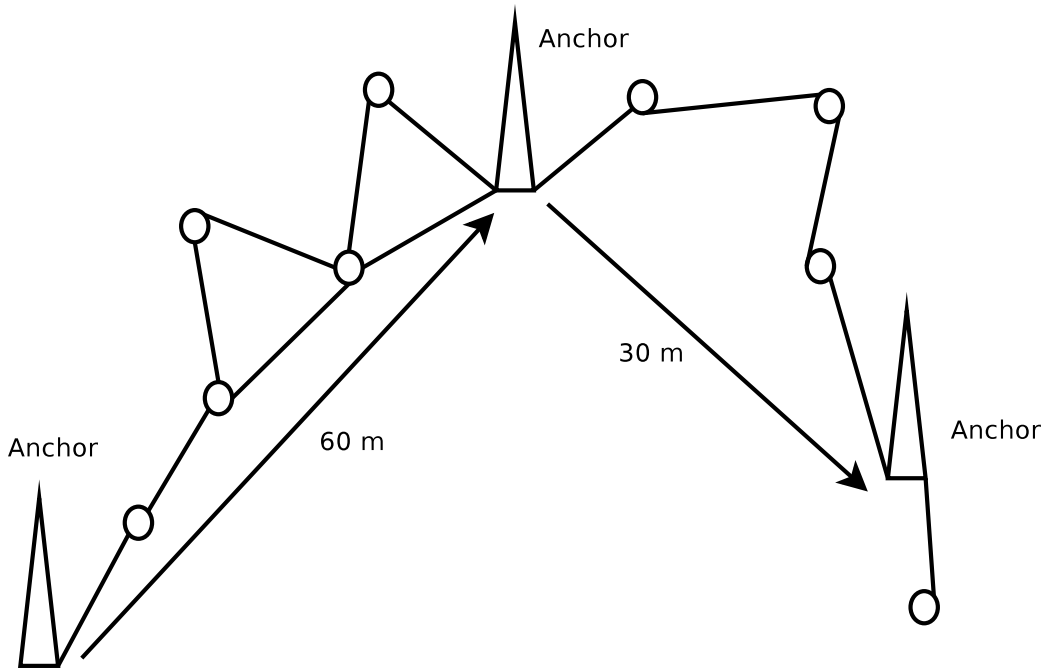


FIG. 2.3. Ad-Hoc Positioning System: multihop localization

Localization measurement is often noisy and incurs some error. Although many good localization algorithms [35][36] [37][38] [39], have been proposed for the wireless ad hoc networks, observations have been made that in all these localization systems, a localization error of 10% of the radio range is very reasonable to assume even for the best existing schemes. This limits the scalability of any design based on location service.

Chapter 3

Routing Protocols for Sensor Networks

In this chapter, we review the routing protocols used in sensor networks. Sensor networks can be considered a special class of ad hoc networks (they are both multi-hop self-configuring wireless networks). Routing for traditional ad hoc networks is a well researched area and it is possible to adapt protocols developed there to sensor network operation. However, several characteristics of sensor networks make such protocols poorly suited for them. These characteristics include: (1) the data dissemination model of the sensor networks is far from the view of the traditional wireless ad hoc networks where mostly the communication is from one source to one destination; (2) data-centric operation where communication is directed to sensors that satisfy a query attributes differs from the IP-centric view of ad hoc networks; and (3) while the emphasis of ad hoc networks is on mobility, many sensor networks are static; and finally, (4) sensor network nodes are more energy constrained than ad hoc nodes, and are not rechargable. Because of these reasons, several new routing protocols have been proposed for sensor networks.

In this chapter, the thesis first introduces the *Stateful* routing protocols which need the routing information maintained at each intermediate node through the data forwarding path. More specifically, state is kept at some nodes about non-local areas in the network (for example, the path to reach some non-local node). We first review an example of such routing protocol that is well widely used ad hoc networks is the Dynamic Source Routing protocol

[14, DSR]. We follow by introducing a simple but effective routing protocol, shortest path (hop-count) routing. The other type of routing protocols that appears better suited for sensor networks is *Stateless* routing. Such protocols only track the position of their neighbors and select among them a neighbor that is likely to be closer to the destination. Examples of this class of protocol include [19, Greedy Forwarding], [18, COMPASS Routing], [15, GPSR][16, GFG]. Some geometric routing without location information proposed recently are introduced [21, 22]. After reviewing Stateless protocols, routing protocol design for real-time applications in sensor networks are overviewed [11, SPEED][13, MMSpeed].

3.1 Stateful Ad Hoc Routing

The stateful ad hoc routing protocols require node to maintain some routing information that is collected using the routing protocol (e.g., through route request propagation or by reversing paths taken by the query). Many routing protocols have been proposed for the traditional wireless ad hoc networks, such as Dynamic source routing (DSR) [14], Ad hoc On-demand Distance Vector Routing (AODV) [40] and Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [41]. DSR and AODV are on-demand routing protocols, constructing routes only when they are needed. In contrast, DSDV is a table-driven routing protocol which proactively constructs routes for all nodes in the network in a similar fashion to traditional wired network routing protocols.

For application on sensor networks, the participants of the routing or communication are mostly of no interest to the user; more often, the sensor are addressed in terms of application dependent attributes (e.g., the sensors in specific region, or detecting a specific phenomena). Thus, traditional IP-centric routing approaches are not well suited to sensor networks. Simplicity and low-cost are main design goals of the routing protocols.

In this section, we briefly describe the DSR to provide a reference point to routing protocols for the traditional wireless ad hoc networks. After that, we describe a simple

routing protocol called shortest path routing for the sensor network application we are interested in.

3.1.1 Dynamic Source Routing(DSR)

The Dynamic Source Routing (DSR) Protocol[14] is a source-routing on-demand protocol. The two major phases of the protocol are: route discovery and route maintenance. When the source node wants to send a packet to a destination, it looks up its route cache to determine if it already contains a route to the destination. Each node maintains route caches containing the source routes that it is aware of. The node updates entries in the route cache as and when it learns about new routes. If the node finds that an unexpired route to the destination exists, then it uses this route to send the packet. On the other hand, if the node does not have such a route, then it initiates the route discovery process by broadcasting a route request packet throughout the network. The route request packet contains the address (usually the IP) of the source and the destination, and a unique identification number. Each intermediate node checks whether it knows of a route to the destination. If it does not, it appends its address to the route record of the packet and forwards the packet to its neighbors. To limit the number of route requests propagated, a node processes the route request packet only if it has not already seen the packet and it's address is not present in the route record of the packet.

A route reply is generated when either the destination or an intermediate node with current information about the destination receives the route request packet. A route request packet reaching such a node already contains, in its route record, the sequence of hops taken from the source to this node. As the route request packet propagates through the network, the route record is formed as shown in figure 3.1. If the route reply is generated by the destination then it places the route record from route request packet into the route reply packet. On the other hand, if the node generating the route reply is an intermediate

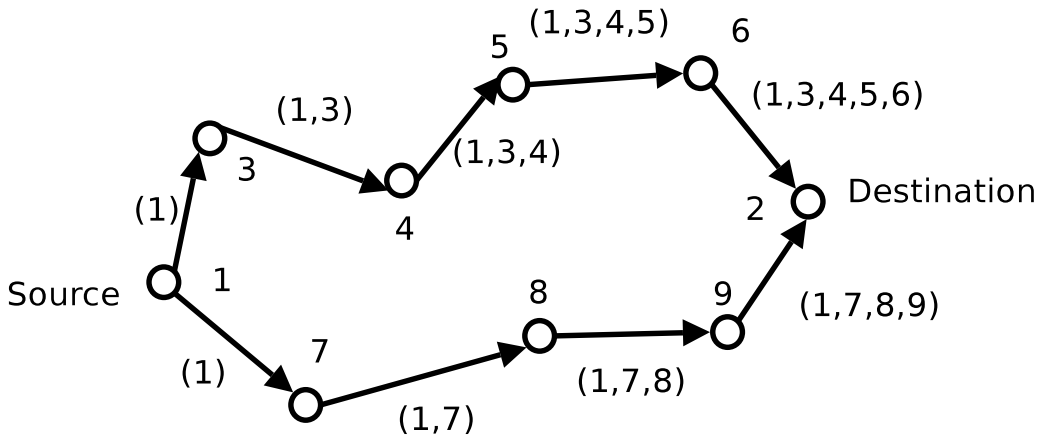


FIG. 3.1. DSR: route discovery

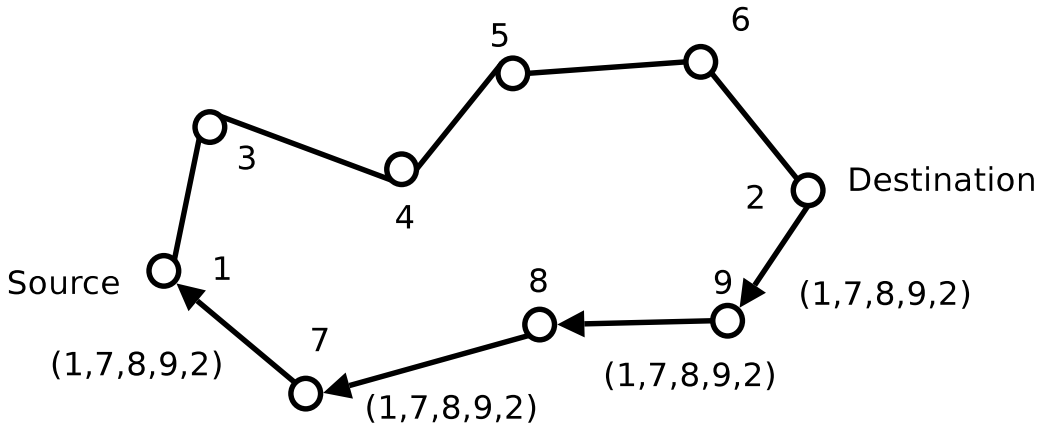


FIG. 3.2. DSR: route reply

node then it appends its cached route to destination to the route record of route request packet and puts that into the route reply packet. Figure 3.2 shows the route reply packet being sent by the destination itself. To send the route reply packet, the responding node must have a route to the source. If it has a route to the source in its route cache, it can use that route. The reverse of route record can be used if symmetric links are supported. In case symmetric links are not supported, the node can initiate route discovery to source and piggyback the route reply on this new route request.

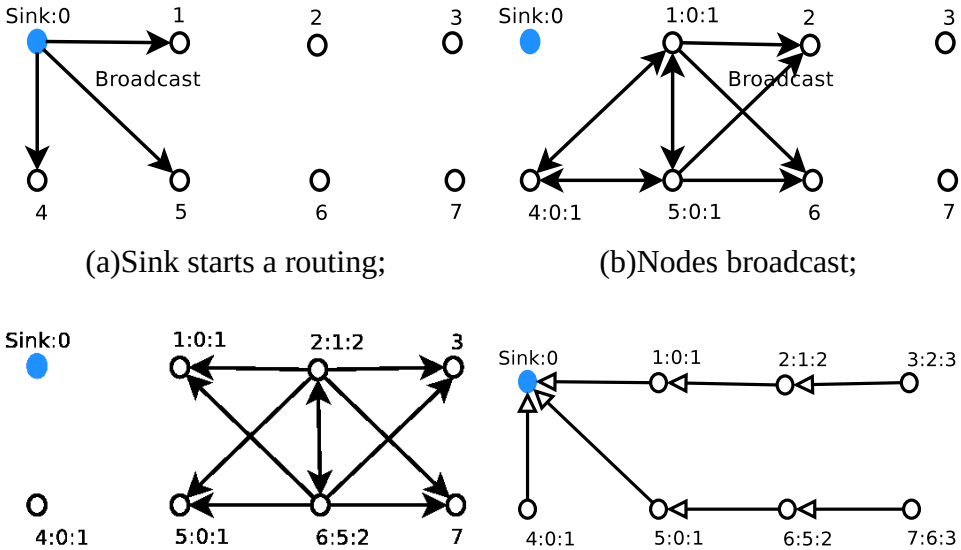
DSR uses Route Error packet to flag invalid links (links that are now unreachable due

to mobility). When a node encounters a fatal transmission problem at its data link layer (when the retransmit limit is reached), it generates a Route Error packet. When a node receives a route error packet informing it of a link that is now unusable, it removes the invalid link from its route cache. All routes that contain invalid link are are truncated to that point.

3.1.2 Shortest Path Routing for sensor networks

The idea of the shortest path routing is based on DSDV[41]. The number of obsevers/data collection points is much smaller than that of the data sources in many sensor networks. Since the data sink is the destination of the data flow, it can be used as the source of triggering routing updates. We call this kind of routing as sink-driven routing. The routing is set up during the interest query broadcasting from the data sink to sources. The data sink broadcasts the query (data request) periodically. A sequence number is used for updating query. The neighbors in turn broadcast the query to their neighbors. On receiving a data query, each node would record the last hop of the query for data packet forwarding in future. The number of hops to the sink (hop-count) of the last hop is also recorded in an incrementing manner. The smallest of the hop-count is used for the greedy route decision in data packet forwarding. An intermediate node broadcasts the query with its own hop-count which is obtained from incrementing the smallest hop-count of all neighbors. A node discards a query with an old sequence number or if the query comes from a neighbor whose hop-count is bigger or the same as the receiver. Essentially, this protocol is similar to AODV with the following distinction: only the sinks propagate the route request packets (in the form of queries). Furthermore, since in general, many nodes are interested in reaching each sink, the query serves to set up routing information for all the nodes concurrently. Further, rather than handling errors as an exception, they are simply tolerated until the next query forwarding period where the routing entries are reset to a valid state.

When a sensor node forwards a data packet, it tries to make a greedy decision to use the neighbor with the smallest hop-count as the forwarding next hop. There may be several neighbors with the smallest hop-count. The node usually chooses the one with the shortest transmission delay. Alternatives are based on different routing policies used, such as choosing the neighbor with the smallest hop-count and one hop delay. To support such a policy the one hop transmission delay of that neighbor would need to be tracked along with other routing information. The possibility of balancing traffic and utilizing multiple routes by distributing the data to multiple candidate neighbors is promising but is not pursued in this thesis. Since the one hop transmission delay may fluctuate due to traffic patterns and interference from other nodes, we just simply the lastest estimate for selecting the next hop selecting: we call this the Estimate of the Transmission Delay (ETD). We estimate the local ETD by exchanging a packet infrequently with the next hop neighbor towards the sink. A more precise estimate of ETD requires MAC layer support. We do not use it because of the difficulty of implementing MAC layer modifications. Moreover, we show that the effect of the resulting inaccuracy is small.



(a)Sink starts a routing; (b)Nodes broadcast;

(c)All nodes receive; (d)Routing Tree

FIG. 3.3. Example: Shortest Path Routing Tree

The routing structure set up towards each sink is essentially a tree rooted at the sink with the matching data-source sensors as the leaves (or intermediate nodes). Figure 3.3 shows an example of the shortest path routing tree for a single data sink. Since the routing decision is made in a greedy manner, the routing tree is usually the minimum spanning tree when considering the network as a graph.

A slightly changed version of the shortest path routing algorithm was recently proposed in [42]. For each data sink, one routing tree needs to constructed throughout the whole network. For this reason, the storage overhead of routing at each node in the network depends on the number of sinks and neighbors. This high overhead may limit the use of such scheme in some large scale networks.

3.2 Stateless Geometric Ad Hoc Routing

Stateful routing may not be efficient or even possible for very large networks with limited sensor node capabilities. Accordingly, **stateless** routing protocols which do not maintain per-route state have been proposed. They scale effectively in terms of routing overhead because the tracked routing information does not grow with the network size or the number of active sinks.

Geographic (and more generally location based) routing protocols are the main type of stateless routing protocols; several examples of these routing protocols exist [15, 16, 18, 20]. They all are stateless routing protocols that track only the location of their one hop neighbors. Recently [17] analyzes the geometric routing behavior in a covered sensor network. For some routing protocols, parameters other than just location, such as the energy efficiency and data aggregation, need to be considered when making routing decisions [43, 44].

This section first describes two types of geometric forwarding routing protocols, analyzing the advantages and shortcoming of both, then briefly describes a popular geometric

routing GPSR [15, 16] which is a hybrid of the first two types, and promises a success in path routing if the path from the source to the sink exists.

3.2.1 Greedy Forwarding (Geographic Forwarding)

Greedy forwarding was first proposed in [19]. The routing decision is purely based on the location of the data sink and forwarding node. Since there is not specialized set-up process required before data can be forwarded, this protocol is gaining popularity for use in sensor networks. The description below is based on the protocol description in [10, 15].

Instead of stamping packets with the ID of the data source and sink, the data packets are marked by the locations of their originator and destination. Any forwarding node makes a locally optimal, greedy routing decision. It divides its all immediate neighbors (the nodes that fall in its transmission range) into two sets: the *forwarding set (FS)* consisting of the neighbors whose distance to the packet's destination is shorter than that of itself; and the other consisting of the rest. The forwarding node chooses the one closest to the packet's destination from the *FS* as the packet's next hop. Forwarding in the regime follows successively closer geographic hops, until the sink is reached. Figure 3.4 shows an example of forwarding decision by greedy forwarding. In this example, when a data packet generated by a source, *X* is the closest one to the packet's sink among all the neighbors of the *source* (including the source itself). This forwarding process is repeated at X and any subsequent hops, until the packet reaches its data sink.

To maintain accurate location information of all neighbors, a simple hello message mechanism is needed. Periodically, each node broadcasts its location with its relatively unique ID (which has to be unique only among two hop neighbors). In GPSR[15], the location information is piggybacked by the MAC layer ACK during the data packet transmission.

The advantage of the greedy forwarding over the stateful routing protocols is just

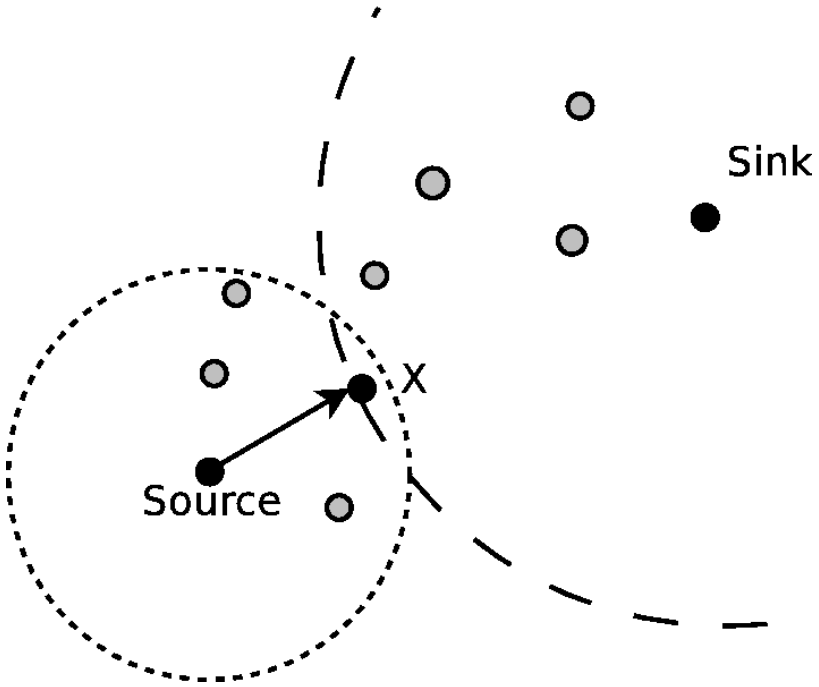


FIG. 3.4. Greedy Forwarding Example: X is source's closet neighbor to Sink

the locations of all its **immediate** neighbors. No matter how many the data sources and sinks are, the size of the routing table in each node keeps fixed, depending only on the network density (how many nodes in one transmission unit averagely, measured in radio range of nodes). The state required is negligible. So this kind of routing protocols are called **Stateless** routing protocols.

Although Greedy Forwarding is effective in many cases, it cannot always find a path to the destination even when such a path exists. There is one kind of topologies in which the greedy forwarding fails [19]. An example is shown in Figure 3.5 where node *X* is a local maximum (also called local minimum in some papers) whose distance to the packet's destination is the shortest one among its immediate neighbors. The dashed curve is part of the circle centered at the *Sink*, whose radius is the distance between *X* and *Sink*. The dotted circles are the ideal radio ranges for nodes. All the immediate neighbors of *X* are farther to the *Sink* from *X* itself. Therefore, *X*'s forwarding set is empty and the packets are dropped.

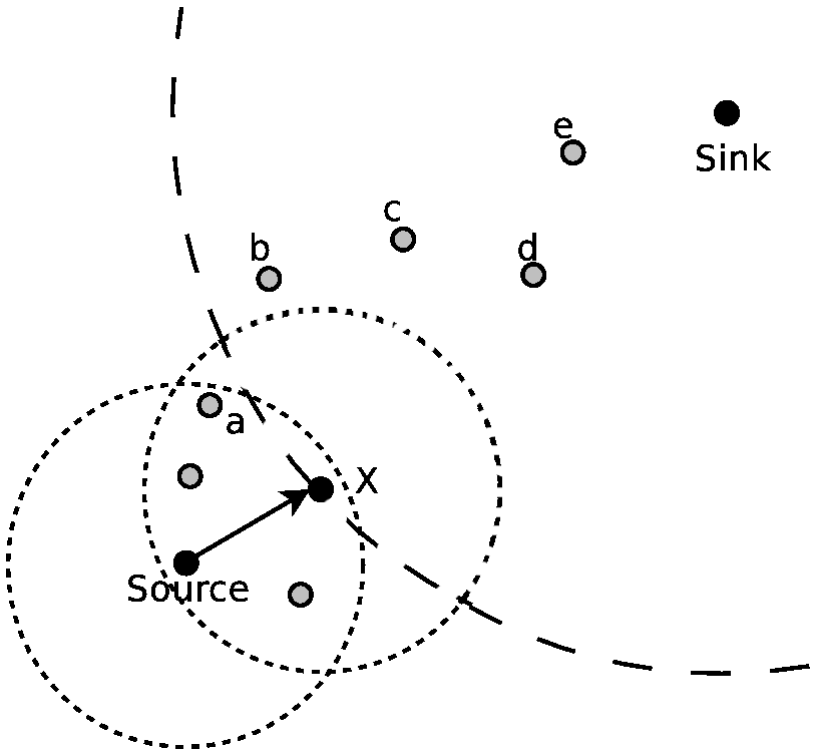


FIG. 3.5. Greedy Forwarding failure: X is a local maximum

Note that a path exists through nodes (source, a, b, c, d, e, sink). As shown in the figure 3.6, the shaded area is called a **Void**.

3.2.2 Face Routing

Face routing is another kind of geometric routing protocol, which is able to overcome the local maximum problem faced by GF. Face routing uses planar graphs to go around voids [15, 16, 18].

Planarized Graph Face routing is based on the theory of *planar graph*. A graph is planar if it can be drawn with no edges that cross each other. It is also called *convexly embedded geometric graph* in [18]. For routing purposes, the sensor network may be viewed as a graph, with nodes as vertices, and edges representing nodes that are in range with each

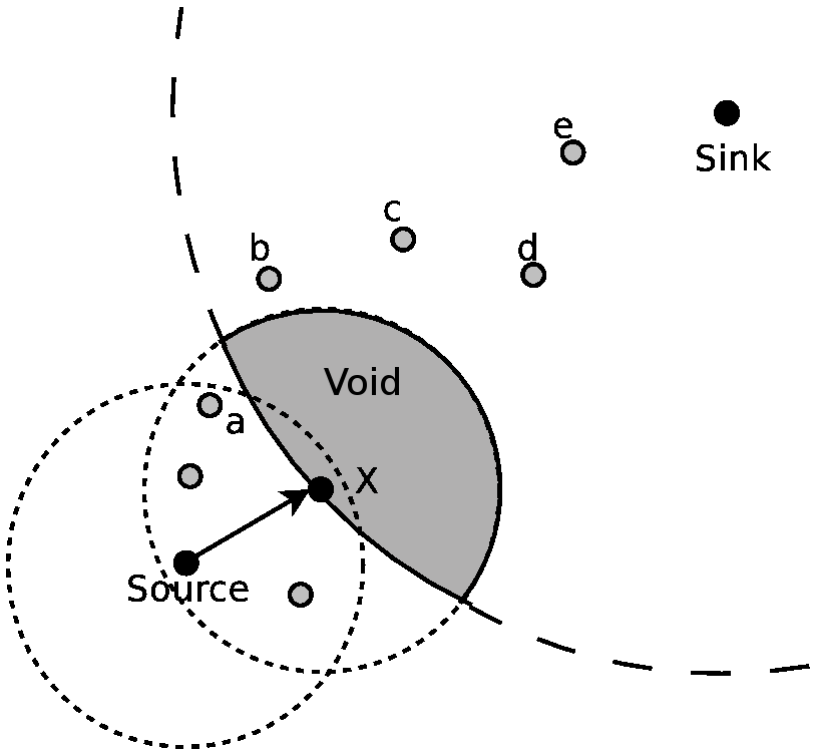


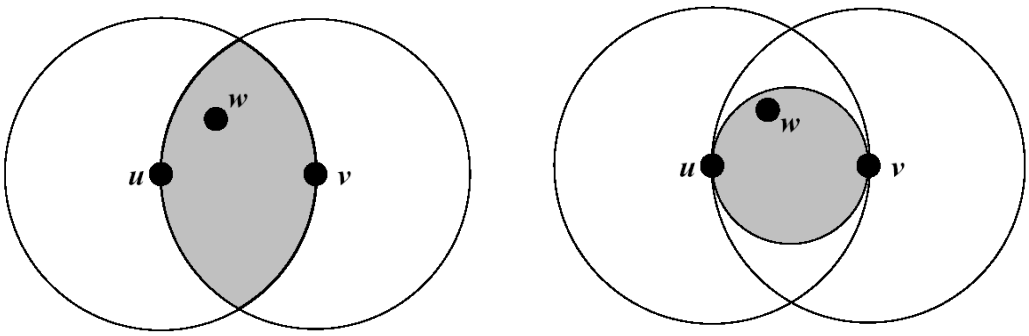
FIG. 3.6. Geometric Routing: Void area for node *X* to *Sink*

other. Most ad hoc networks are not planar. However, if we remove the edges that cross each other, the graph becomes planar. Face routing protocols employ some algorithms to remove the crossing links distributedly. Any algorithms should not disconnect the graph during removing edges.

The *Relative Neighborhood Graph* (RNG) and *Gabriel Graph* (GG) are two kind of planar graphs. Figure 3.7 shows examples for RNG and GG planar graphs.

Definition RNG: An edge (u, v) exists between vertices u and v if the distance between them, $d(u, v)$, is less than or equal to the distance between every other vertex w, and whichever of u and v is farther from w. In equational form:

$$\forall w \neq u, v : d(u, v) \leq max[d(u, w), d(v, w)]$$



RNG example GG example

FIG. 3.7. Edge(u, v) belongs to a RNG/GG planar graph if there exists **NO** w.

Definition GG: An edge(u, v) exists between vertices u and v if no other vertex w is present within the circle whose diameter is $\overline{uv}$. In equational form:

$$\forall w \neq u, v : d^2(u, v) < [d^2(u, w) + d^2(v, w)]$$

It has been shown in literature that the RNG is a subset of the GG [45]. To remove the cross links, each sensor node eliminates the links to neighbors where there exists a witness node w. After removing all the cross links according above definitions, the network consists of multiple convex polygons called faces.

Face routing based on planarized networks Here we repeat an example from Compass Routing II[18] to show how a face routing works. Figure 3.8 shows a routing based on the *Compass Routing II* which is the original version of face routing. The routing algorithm would do

1. Starting at s determine the face $F = F_0$ incident to s intersected by the line $\overline{st}$. Pick any of the two edges of F_0 incident to s, and traversing the edges of F_0 until it finds boundary edge of F_0 intersected by $\overline{st}$ at point p_0, shown as in the figure 3.8;

2. At the intersected point p_0, it changes the face $F = F_0$ to $F = F_1$. Subsequently, it

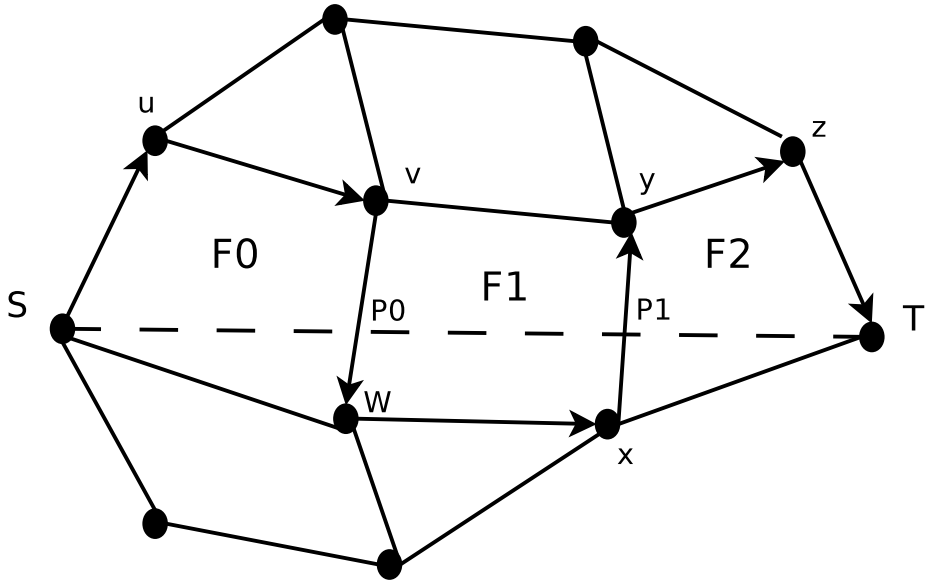


FIG. 3.8. Face routing example: Compass Routing II

repeats the previous step until a new intersected point p_1 found;

3. Suppose there is a sequence of faces $F_0, F_1, \cdots, F_n$ intersected by the line $\overline{st}$. The previous steps would be repeated $n + 1$ times until the destination t is reached. In Figure 3.8, the $n = 2$.

Face routing is not an efficient routing protocols. It sacrifices some communication efficiency for routing success, which leads to lower throughput of the networks. It may generate much longer path than a greedy algorithm. So mostly, the performance of a pure face routing protocol is much worse than a greedy one.

3.2.3 GPSR: Greedy perimeter Stateless Routing

GPSR[1]. It is a hybrid protocol which combines Greedy forwarding with face routing. By default, the protocol employs greedy forwarding. However, if a local maximum is reached, face routing is then used to route the packets around the voids.

[1]Although GPSR[15] is a duplication of GFG[16] and published one year later, it is more widely cited.

Upon receiving a greedy-mode packet for forwarding, a node searches its neighbor table for the neighbor geographically closest to the packet s destination. If this neighbor is closer to the destination, the node forwards the packet to that neighbor, which is the same behavior of greedy forwarding. When no neighbor is closer (a local maximum situation happens), the node marks the packet into perimeter mode (face routing mode). As a result, the data packet is forwarded in a face routing mode until: (1) the destination is reached; or (2) a node closer to the destination is reached than the node where the packet enters the face-routing (or perimeter) mode. During the perimeter forwarding, the next forwarding edge is chosen based on a right-hand rule (shown using an exampl ein Figure 3.9).

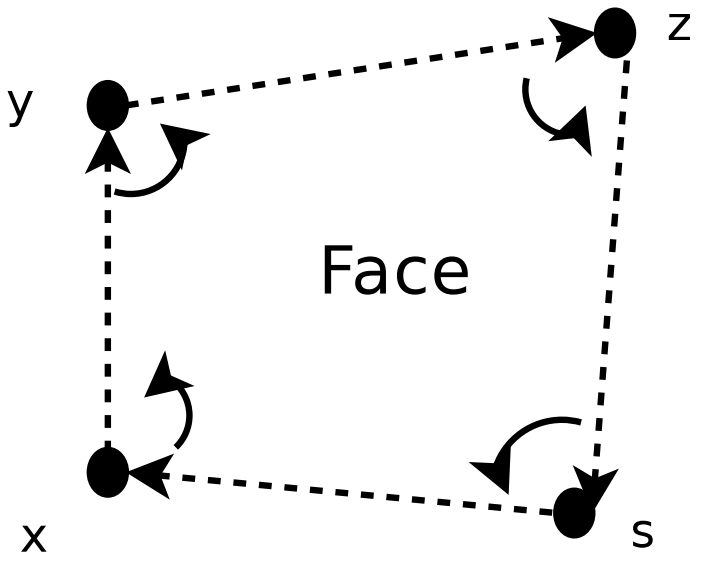


FIG. 3.9. Right-hand Rule: y receives a packet from x, and forwards it to its first neighbor counterclockwise about itself z.

Another overhead of perimeter forwarding is face caching: the edges of the same face traversed by the data packets need to be record in the data packet header. In addition, the network graph needs to be planarized before a data packet could be perimeter forwarded. Although the overhead seems to be a little high, the proportion of the perimeter forwarding is small in well connected networks.

3.2.4 The Impact of Localization Errors

Although the stateless geometric routing is attractive it is critically dependent on the accuracy of the localization algorithm If the location information is erroneous forwarding decisions that rely on them may not be able to lead the packet towards the destination. An analysis of the impact of the localization error on the greedy forwarding was shown in [37]: for average localization errors of 20% and 40% of the node radio range, the delivery ratios of greedy forwarding are very close to the baseline (no error). However, our simulation disagree with this result: even when the error is just in 10% of the radio range, the impact on the delivery ratio is drastic in some cases. A recent paper [46] presented a detailed micro-level analysis of pathologies for geographic face-based routing protocols, in the presence of location errors in static sensor networks. It also shows that the localization error affects the face routing performance extensively.

3.3 Other Routing Protocols

Recently, some geometric routing protocols that do not use location information have been proposed [21, 22, 47]. In [21] a node uses the average coordinate location of all neighbors as it location. All nodes initialize own location as the center coordinates of a area. It takes a long time (nearly 1000 iterations) for the node locations to converge to accurate values. In [22, 47], a virtual coordinates system that assigns coordinates to nodes based on hop-count is introduced. However, this approach may lead to multiple nodes being labeled with the same coordinates (nodes that are equally distant from the anchors); this problem is called *aliasing*. Besides aliasing, both of protocols are designed just for greedy forwarding; no mechanisms for navingating voids are provided. Although the [47] provides a solution to both the problems of aliasing and void, the cost of the solution which is based on broadcast search is prohibitive.

3.4 Routing protocols under Real-time Constraints

Meeting real-time constraints in a sensor network requires careful design of both routing protocol and packet scheduling policies. Traditional routing protocols reviewed thusfar are not sensitive to real-time traffic. There is a need for new routing protocols that can better support real-time applications on sensor networks.

3.4.1 SPEED

SPEED[11, 12] is a well-known stateless routing protocol for real-time communication in sensor networks. It is based on geometric routing protocols such as greedy forwarding[19] and GPSR[15, 16]. It uses non-deterministic forwarding to balance each flow among multiple concurrent routes.

In the design of SPEED, the end-to-end deadline is reflected as the data packet forwarding "speed". The end-to-end distance is measured in Euclidean distance. Speed, where $speed = \frac{E2E\ distance}{E2E\ deadline}$, is used as a parameter to make routing decision. The node generating the data packet would put the required "speed" (set point) into the packet header. When a node receives a data packet with this required "speed", it uses the local information to make real-time routing decisions. The location information of all neighbors is maintained at each node, as well as the one hop transmission delay of each neighbor. The routing decision is made as follows: Only the neighbors whose distance to the destination is shorter than the forwarding node are possible forwarding candidates (this group of neighbors is the FS); the forwarding candidate in FS with the highest one hop "speed" would be the next hop of the data packet. The distance used to calculate the one hop "speed" is the progress distance shown in figure3.10. Back-pressure re-routing is used when there is no neighbor in FS able to maintain a one-hop "speed", which may cause the back-pressure rerouting. Since SPEED is based on the greedy forwarding routing, it suffers from the Void problem of greedy forwarding. The node with no greedy next hop would drop the incoming

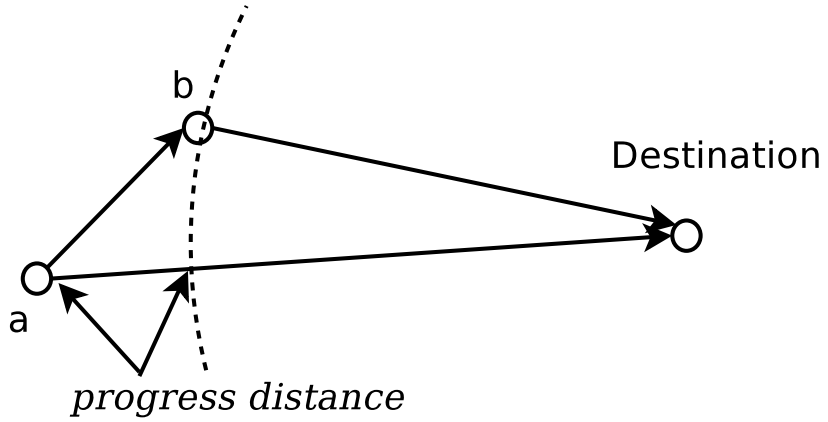


FIG. 3.10. SPEED: progress distance

data packet, and back-pressure re-routing is used to update the routing in such cases, which also hurts the performance.

3.4.2 MMSpeed

MMSpeed[13] is an extension of SPEED[11] that focuses on differentiated QoS options for real-time applications with multiple different deadlines. It provides differentiated QoS options both in timeliness domain and the reliability domain. For data flows with different end-to-end real-time constraints, MMSPEED provides a multi-layer mechanism to differentiate these flows. Each layer is dedicated to a specific required ”speed”, including both routing and MAC mechanisms. A FCFS queue is used for each layer. To differentiate data flows with different reliability, the multipath routing protocols are used. Load balance is taken care of well by this kind of routing.

A small modification may resolve this problem in the ”single layer” scheduling algorithm. Multi-layer mechanism for different data flows with varying end-to-end ”speed” or deadline requirements seems unecessary. Despite of that, the implementation of the multi-layer mechanism is costly since it requires a complex implementation both at the routing and MAC layers. In a heavy traffic environment, the multi-path routing does not help much

since the traffic ”density” may be common every where. This may also increase the offered load to the network further worsening the situation Another limitation of it is the ”speed” idea borrowed from the SPEED. Although the geometric routing is very scalable, dedicating to it still harms the scalability of SPEED, since for some systems, the geometric routing is not adaptable (for example, the localization service is not available).

Chapter 4

Scheduling for Real-Time Sensor Network Applications

In this chapter, we define the real-time scheduling problem and discuss the challenges involved. In addition, we review Velocity Monotonic Scheduling (VMS) and its use in the RAP [10] framework. RAP is a widely used real-time data dissemination framework; since we compare the performance of our scheduling framework against RAP, it is important to describe its implementation.

The goal of real-time scheduling algorithms is to ensure that critical timeliness constraints, such as deadlines and response time, are met. When necessary, decisions are made that favor the most critical timing constraints, even at the cost of violating others. Real-time scheduling is also used to allocate processor time between tasks in soft real-time embedded systems. We do not pursue real-time scheduling for task processing on sensor networks; rather, we focus on communication scheduling.

Traditional real-time scheduling algorithms are designed for centralized systems such as a mainframe. A simple model of the function of a scheduler is shown in Figure 4.1. For an input set of paramter values of a specified job, the scheduler outputs the priority of this job that is used for arbitrating scheduling on a critical resource.

In the applications based on sensor networks, the scheduling algorithms have to work in a distributed manner. The scheduler residing on each node makes scheduling plans without any global knowledge. The challenge here is how to schedule multiple data packets

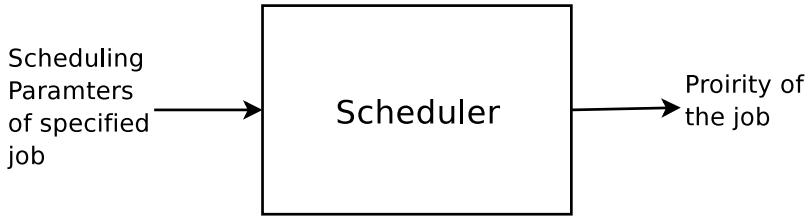


FIG. 4.1. Scheduler function model

at each node independently so as to minimize the global data transmission miss ratio. Li et al [48] prove that the problem of optimal scheduling of parallel messages with deadlines over wireless channel is NP-hard; heuristic solutions are needed.

In this chapter, we first examine a typical classical real-time scheduling algorithm, Rate Monotonic Scheduling (RMS), for centralized systems. We then analyze the real-time data traffic behavior in a wireless sensor networks. Finally, we compare the Velocity Monotonic Scheduling (VMS) and our Just-in-Time Scheduling (JiTS).

4.1 Rate Monotonic Scheduling

Rate Monotonic Scheduling (RMS) [49] is a classic hard real-time CPU scheduling algorithm. It was designed for processors to schedule incoming tasks. It was implemented in RT-Mach using processor capacity reserves, which reserves, for a given task, a requested amount of CPU time during each of its periods [50]. RMS is a simple, preemptive, fixed-priority scheduler, which assigns the resource to the job with the highest priority that is in need of the resource. It works through admit/reject and priority assignment decisions. All admitted jobs are guaranteed to meet their deadlines through a priority assignment that is rate monotonic. Jobs that are not admitted may be either discarded, or allowed to execute at a priority lower than that of all admitted jobs.

To clarify these ideas, we show an example of RMS from [51]. The RMS procedure assigns fixed priorities to tasks to maximize their ”schedulability”. A task set is considered

schedulable if all tasks meet all deadlines all the time. The algorithm is simple:

> *Assign the priority of each task according to its period, so that the shorter the period the higher the priority.*

Let us consider a system with only two periodic tasks: Task 1 and Task 2 with periods T_1 and T_2 respectively. Each task has a deadline that is the beginning of its next cycle. Task 1 has $T_1 = 50ms$, and a worst-case execution time of $C_1 = 25ms$. Task 2 has $T_2 = 100ms$ and $C_2 = 40ms$. Note that the utilization, U_i, of task i is C_i/T_i. Thus $U_1 = 50\%$ and $U_2 = 40\%$. This means total requested utilization $U = U_1 + U_2 = 90\%$. It seems logical that if utilization is less than 100%, there should be enough available CPU time to execute both tasks.

Consider a static priority scheduling algorithm. With two tasks, there are only two possibilities:

- Case 1: Priority(t_1) > Priority(t_2), seen in figure 4.2
- Case 2: Priority(t_1) < Priority(t_2), seen in figure 4.3

The two cases are shown below. In Case 1, both tasks meet their respective deadlines. In Case 2, however, Task 1 misses a deadline, despite 10% idle time. This illustrates the importance of priority assignment. In the example, the period of Task 1 is shorter than the period of Task 2. Following RMS's rule, we assign the higher priority to Task 1. This corresponds to Case 1 in figure 4.2, which is the priority assignment that succeeded in meeting all deadlines.

RMS is the optimal static-priority algorithm. If a task set cannot be scheduled using the RMS algorithm, it cannot be scheduled using any static-priority algorithm.

A major limitation of fixed-priority scheduling is that it is not always possible to fully utilize the CPU. Even though RMS is the optimal fixed-priority scheme, it has a worst-case

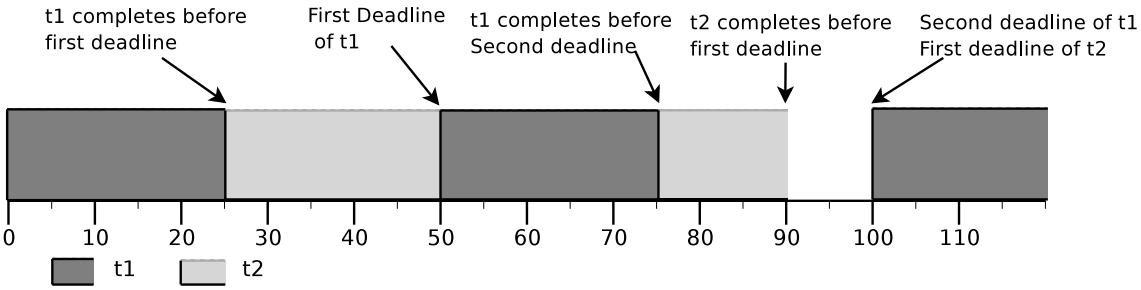


FIG. 4.2. RMS example, Case 1: Priority(t_1) > Priority(t_2)

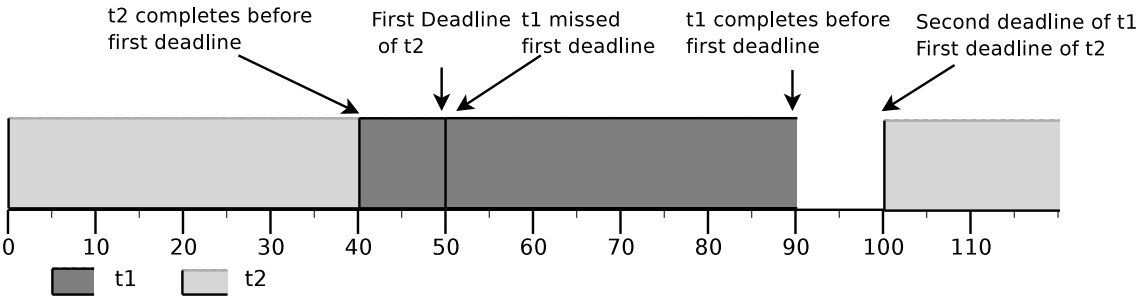


FIG. 4.3. RMS example, Case 2: Priority(t_1) < Priority(t_2)

schedule bound of:

$$W_n = n * (2^{\frac{1}{n}} - 1)$$

where n is the number of tasks in a system. The worst-case schedulable bound for one task is 100%. However, as the number of tasks increases, the schedulable bound decreases, eventually approaching its limit of about 69.3% ($ln2$, to be precise). It is theoretically possible for a set of tasks to require just 70% CPU utilization in sum and still not meet all their deadlines. For example, consider the case shown in the figure 4.4. The only change is that both the period and execution time of Task 2 have been lowered. Based on RMA, Task 1 is assigned higher priority. Despite only 90% utilization, Task 2 misses its first deadline. Reversing priorities would not have improved the situation. In this case, the only way to meet all deadlines is to use a dynamic scheduling algorithm, which, because it increases system complexity, is not available in many commercial Real-Time Operating

Systems (RTOS).

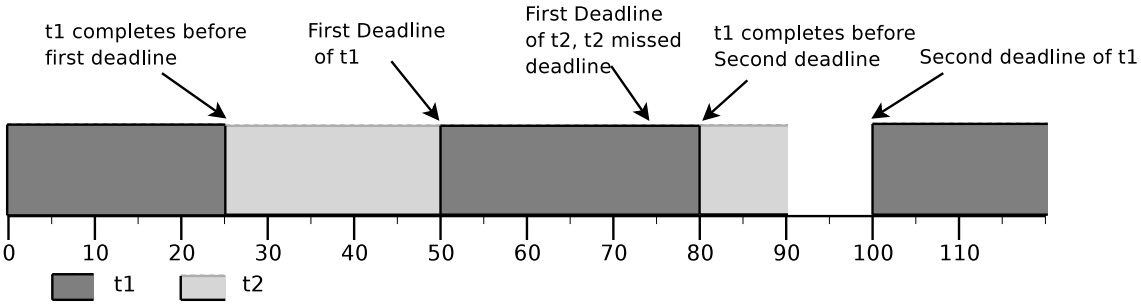


FIG. 4.4. RMS Example, Case 3: unschedulable task sets

It is possible for a set of tasks to have total utilization above the worst-case schedulable bound and still be schedulable with fixed priorities. Schedulability then depends on the specifics of the periods and execution times of each task in the set.

4.2 End-to-End Delay due to Packet Network Delay

In the wireless sensor networks, the delay experienced by real-time data traffic is different from the typical delay experienced by tasks in a centralized system (which are typically considered in real-time scheduling problems). For real-time data dissemination in a wireless ad hoc network, the problem is providing timeliness guarantees for multi-hop transmission. Any data packet associated with a deadline may need to traverse multiple hops from the source to the sink. In [48] it is shown that the problem of meeting message (data packet) deadlines is **NP-hard** even for single hop message transmission. Therefore, heuristic algorithms musat be used to schedule the data packets in a multi-hop ad hoc network .

In a distributed system such as a sensor network where decisions at different nodes interfere with each other (e.g., by keeping the channel busy and preventing other nodes from transmission), traditional scheduling algorithm such as RMS can not be directly used. A

first attempt at a solution to real-time scheduling may consider that minimizing the packet delay at every node will lead to the best chance at meeting the collective real-time deadlines. If the system is centralized, and packet data generation is predictable effective scheduling in this manner can be directly carried out. However, we will show that **minimizing the transport time of each packet does not lead to an optimal solution**. The transport time of a data packet from the source to sink is influenced by the propagation delay, but more importantly the queueing delay at intermediate nodes. The queueing delay is due to contention for the use of the channel and/or routing imbalance leading to congestion in certain parts of the network. In the remainder of this section, we break down the delay experienced by packets in a sensor network.

4.2.1 Transmission and Processing Delay

The propagation delay of a packet is determined by the bandwidth and the size of the message. In a multi-hop wireless network, when each intermediate node along the path to the sink receives a packet, they have to be classified and forwarded. Moreover, for some applications, some application specific processing may also be necessary. This processing time may be considerable, in which case real-time scheduling of packet processing is needed. For data packets generated by different applications, or even different instances of data packet generated by the same application, the processing would be different. In this thesis, we do not consider processing delay separately; rather we combine it with transmission delay as the minimum delay of the packet at the intermediate node. This component of the delay is the minimum delay a packet can spend at an intermediate node.

4.2.2 Queueing Delay

The **Queueing Delay** is the time spent in the queues at each intermediate nodes on the path from the source to the sink. As a packet is received at an intermediate node, it is

placed in the MAC layer incoming queue (a physical layer queue is possible before). When the network layer receives from the queue, it looks at the packet and decides where it needs to be forwarded next. The routing layer places the packet in a local queue while making routing decision. Consequently, the packet is placed in the MAC layer outoging queue to be forwarded to the next hop. Since a queue is used to store a data packet temporarily at any forwarding node, the queueing delay may be controlable by a scheduler.

For different systems and applications, the processing time can be different (across applications or within an application). Moreover, the packet generation activity may fluctuate – for example, an improtant event may occur, leading to the generation of data at a high rate for a period of time. Accordingly, the time spent in different queues, and the length of the queues changes over time: the queueing delay may grow to be the main part of the end-to-end packet transport time especially under high load. More specifically, under moderate to high data generation rates, due to the channel contention, congestion and other reasons, the queueing contribution to delay can far outgrow the physical transmission and processing contributions. However, this part of time in the delay has been ignored in most existing real-time scheduling protocols in sensor networks. To address this shortcoming, the proposed scheduling algorithm is aware of, and attempts to compensate for queueing delay. By considering this important component of delay, which becomes increasingly more important as the load on the system increases, the global miss ratio can be signficiantly improved.

4.3 Deadline-based Scheduling

RMS is a deadline-based scheduling algorithm for centralized system. For a distributed system, the scheduling decision has to be made distributedly. In a wireless sensor network, each sensor node does this job independently. The deadline-based scheduling procedure schedules all incoming data packets just based on their deadlines. If a data

packet has an earlier deadline, it receives a higher priority than another with later deadline. These scheduling algorithms do not differentiate the up-to-date real-time status of each packet through its forwarding path and therefore may hurt the overall performance of the real-time scheduling system. Let's consider the example shown in the figure 4.5. In

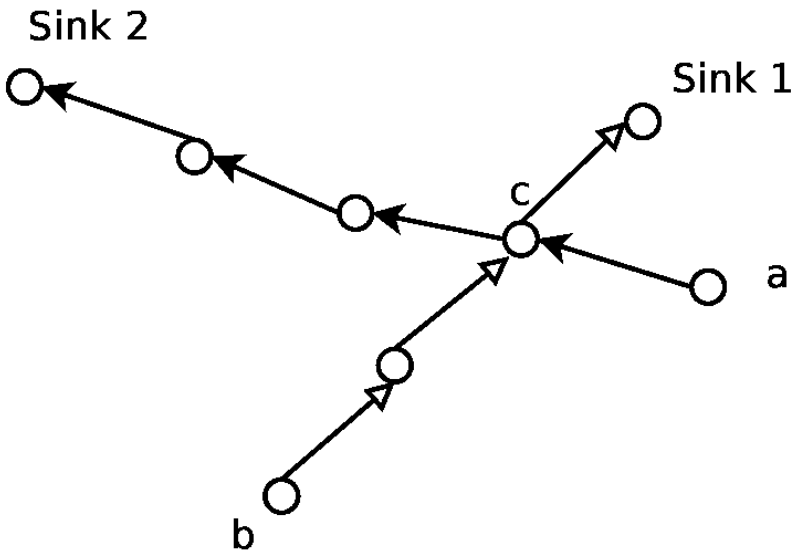


FIG. 4.5. Deadline-based Scheduling example

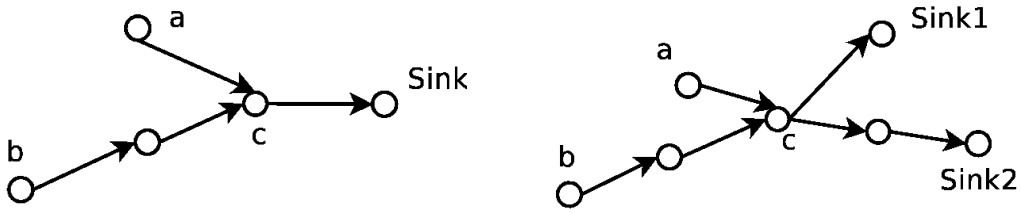
this example, suppose that a data packet generated by node a, p_a, and another generated by node b, p_b arrive at node c at the same time. Both of these two data packets were assigned the same deadline period (the time from originating to expiring). There are possible two scheduling policies employed:

- Static scheduling: only the end-to-end deadline periods of packets are used for assigning priority;
- Dynamic scheduling: the remaining time of the deadline period is used for priority assignment. This means that the time from arriving the current node to deadline expiration is used to determine priority.

For the static scheduling, p_a and p_b would be assigned the same prioirity at any forwarding node. For the dynamic scheduling, since the p_b has spent more time before it reaches node c than p_a, it would receive a higher priority than p_a. However, p_b has only one hop to traverse before it reaches the sink while p_a has 3 hops: p_a is more likely miss its deadline if the remaining time is close. As can be seen, both static and dynamic deadline-based scheduling are not able to support p_a's deadline in this scenario. Thus, a scheduling algorithm should not only be aware of the remaining slack time, but also of the expected remaining delay that the packet needs to reach its destination.

4.4 Velocity Monotonic Scheduling

Velocity Monotonic Scheduling (VMS) was first proposed by He et al [10, RAP], which was based on the idea of *multi-hop coordinated scheduling*[52].



(a) Node c schedules the packets from different sources a and b with different distances to the same sink

(b) Node c schedules packets from different sources to different sinks with the same distances

FIG. 4.6. Velocity Monotonic Scheduling example

The key role of a scheduling algorithm in a real-time communication architecture is to schedule the incoming packets for forwarding. Traditional FCFS scheduling does not work well in real-time applications because packets have different remaining times to their deadlines, but also different distances to reach their destination. Instead of FCFS, the

VMS prioritizes the competing packets based on their perceived urgency (called *velocity*). Velocity is determined as the ratio of the deadline time to the distance that the packet has to travel: the intuition is that packet scheduling should balance the available time to deadline against the distance that the packet has to travel[10]. VMS improves the real-time scheduling since it assigns priorities with consideration to the expected remaining delay (as approximated by distance to sink), and not just the deadline. Figure 4.6 shows 2 different scenarios that can be satisified by VMS scheduling. There are two flavors of VMS scheduling: static and dynamic.

4.4.1 Static Velocity Monotonic Scheduling (SVM)

Static VMS (SVM) prioritizes packets based on a fixed requested velocity that is computed at packet generation time. The values of parameters used by prioritizing at any intermediate node is that of the data source, not the forwarding node. So the velocity is fixed or static and defined as follows:

$$Velocity = \frac{distance(source, sink)}{E2E\ Deadline}$$

For routing protocols with different policies, the distance between source and sink is measured in different ways. For example, for shortest path routing, the distance is measured in number of hops; for geometric routing, it would be measured in Euclidean distance. The shortcoming of SVM is that since the delay times of packets at intermediate nodes are unpredictable, it is unable to increase the priority of a packet that is unexpectedly delayed, or reduce the velocity of a packet that is lucky enough to make quick progress initially towards the destination.

4.4.2 Dynamic Velocity Monotonic Scheduling (DVM)

Dynamic VMS (DVM) recomputes packet velocity at each intermediate node. For a packet from a source whose distance is d, urgency is not only based on the end-to-end deadline, but also the time is already elapsed. Or we can consider the *time left to deadline*. The value of velocity is decided by function:

$$Velocity = \frac{distance(forwarding\ node, sink)}{E2E\ Deadline - Elapsed\ Time}$$

In Figure 4.6, the velocities of packets from node a and b for scheduling at node c are shown in table4.1.

Packet	SVM velocity	DVM velocity
p_a	$\frac{dis_a}{dl}$	$\frac{dis_a}{dl - T_a}$
p_b	$\frac{dis_b}{dl}$	$\frac{dis_b}{dl - T_b}$

Table 4.1. VMS example:

Consider scenario (a) first. Node c attempts to schedule a data packet from node a, p_a, and one from b, p_b, with the same end-to-end deadline dl. Suppose the distance of a to sink dis_a is shorter than that of node b to sink dis_b. If SVM is used, packet p_b receives a higher priority than p_a. This makes sense since the p_b needs to traverse a longer distance with the same deadline as packet p_a. In general, when packet p_a arrives at node c, T_a is smaller than T_b which means the left time to deadline of packet p_a is longer than that of packet p_b. So if DVM is used, packet p_b receives a higher priority in DVM. In scenario (b), the forwarding node c tries to schedule p_a and p_b with the same deadline dl. Suppose $dis_a = dis_b$. If SVM is used, packet p_b will receive the same priority as p_a. If DVM is used, since the T_b is longer than T_a, the p_b will receive a higher priority than p_a.

4.5 RAP: a Real-time Architecture for Sensor Networks

RAP[10] is a real-time framework developed at the University of Virginia. Since we use RAP as the baseline real-time framework against which we compare our framework, in this section we describe the RAP implementation in detail. RAP uses Velocity Monotonic Scheduling (VMS) to prioritize the data packets which was inspired by [52]. Three queues are used for scheduling packets per the VMS algorithm described earlier in this chapter. Packets fall in one of the three different priority level and are queued into the corresponding queue in a FIFO manner. RAP does not consider the queueing delays of data packets; once the packet is assigned a priority it remains at that priority regardless of queueing delay. RAP also modifies the MAC layer back-off schemes to schedule packet transmission.

4.5.1 Other Related Work

The SPEED framework from the same group that developed RAP[11][12] proposed an optimized Geographic Forwarding routing algorithm for sensor network. SPEED uses a MAC layer estimate of one-hop transmission delay to select the next hop to forward the data packet to. SPEED does not queue the data packets by scheduling or even prioritize the data packets.

Li et al [53] prove that the problem of scheduling parallel messages with deadlines over wireless channel is NP-hard. They propose Last Start Time First (LSTF) scheduling which schedules the messages based on their per-hop timeliness constraints (using manipulation of MAC layer backoffs). The paper also studied the spatial reuse of the wireless channel as well as the effect of collision avoidance.

Multi-hop Coordination riority Scheduling citepaper:dpsch is posed to incorporate distributed priority scheduling into existing IEEE 802.11 priority back-off schemes to approximate an ideal schedule. This scheme allows the downstream nodes to increase a packets relative priority to make up for excessive delays incurred upstream. The scheduling

requires complex modifications to the MAC layer.

Clearly, the ability to meet real-time deadlines in the presence of contention is related to controlling the load presented to the system. In terms of networking resources, this is the problem of congestion control. The importance of congestion control in sensor networks was identified [54] and approaches for addressing it have been developed [55]. Kang et al study the possibility of addressing congestion by using multiple paths[56]. Exploring the intersection of congestion control and real-time scheduling is an interesting topic of future research.

Also related to real-time support is Quality of Service support. An example of QoS frameworks for ad hoc networks is SWAN[57]: a stateless QoS framework supporting service differentiation for real-time and best-effort traffic. SWAN supports perhop and end-to-end control algorithms without per-flow information. It uses local rate control for UDP and TCP best-effort traffic, and sender-based admission control for real-time UDP traffic. Explicit congestion notification (ECN) is used to dynamically regulate admitted real-time sessions in the face of network dynamics that arise from mobility or traffic changes. While the network topology in ad hoc networks is similar to that of sensor networks, SWAN and similar models target unicast network traffic and do not map to sensor networks many-to-1 data dissemination operation.

4.5.2 The RAP mechanism

The main design goal of RAP is to develop a fair scheduling algorithm for data dissemination in a distributed wireless sensor network, so as to minimize the miss ratio of real-time data traffic. RAP requires support from several layers in the network, which represents a significant deployment barrier, especially for heterogeneous networks. The main contribution of RAP is the *Velocity Monotonic Scheduling* (VMS), described in Section4.4.

Velocity Monotonic Scheduling with Greedy Forwarding The choice of routing protocol in RAP is *Greedy Forwarding* (GF), a simple geometric routing described in Section3.2.1. Since GF routing cannot resolve the void problem (seen an example in figure3.6), for some realistic deployment of sensor network, more sophisticated face routing algorithms are required in the event that GF reaches a local maximum. However, the VMS scheduling algorithm assumes underlying GF operation since the distance parameter of VMS assumes straight line forwarding towards the sink as shown in Figure4.7. Assume

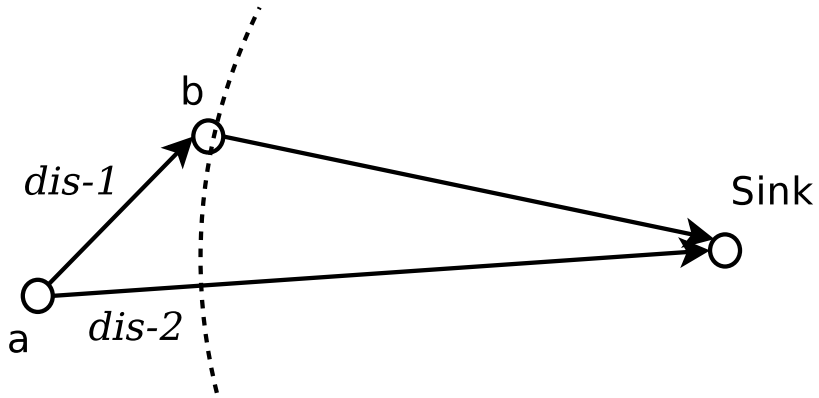


FIG. 4.7. The distance parameter used by VMS in RAP

node a generates a data packet p_a to the sink. Among a's immediate neighbors, b offers the shortest distance to the sink. Accordingly, a forwards p_a to b based on the priority assigned by VMS scheduling. The priorities of p_a according to different scheduling policies at node a and b shown in table4.2. As we have seen, static VMS assigns the same priority to a data packet at each hop and then schedules the data packets based on their end-to-end requested velocity, ignoring their current status. Dynamic VMS uses an up-to-date requested velocity of data packet to make scheduling decision. However, the distance parameter value used for intermediate node scheduling is not as fair as in SVM. When p_a arrives at node b, it traversed a distance dis_1 shown in Figure4.7 within the *Elapsed Time*. But

$$distance(b, sink) = distance(a, sink) - dis_2 \neq distance(a, sink) - dis_1$$

VMS policy	At node a	At node b
Static VMS	$\frac{distance(a, sink)}{E2E\ Deadline}$	$\frac{distance(a, sink)}{E2E\ Deadline}$
Dynamic VMS	$\frac{distance(a, sink)}{E2E\ Deadline}$	$\frac{distance(b, sink)}{E2E\ Deadline - Elapsed\ Time}$

Table 4.2. Packet priorities when different VMS policies are used

The situation is even worse with as the number of hops the packet traverses from source to sink increases.

Priority Queue In RAP, each incoming data packet is assigned a priority by VMS scheduler based on its requested velocity, then queued into the priority queues before it is forwarded. Instead of a single priority queue for all packets, RAP uses multiple FIFO queues each corresponding to a fixed priority level. Each priority level corresponds to a range of the requested velocity. A packet would be appended to the end of a queue if its priority falls into the range binding to this queue. The reason to use multiple FIFO queues is to enhance the efficiency of the system, avoiding the per-packet overhead of priority queue operation. The choice of the priority ranges of queues is a key problem of this method.[1]

MAC layer Support The RAP designers observe that since access to the wireless network is not prioritized, local prioritization of packets may not prevent priority inversion when considering transmissions from multiple senders on the shared radio channel. RAP uses the IEEE 802.11 wireless LAN protocol[27] as its MAC layer protocol. To prioritize the packets in the MAC layer when they compete for the communication channel, it adapts an extension of the IEEE 802.11 protocol proposed in[58]. The packets with higher priority

[1]The authors did not answer this question in the paper despite the fact that this choice significantly affects the performance of RAP.

would win the channel more often using this modification. Priority is introduced in two ways: by having higher priority packets wait less before initial transmission and backoff less in the event of a collision. We describe these modifications briefly below.

Initial Wait Time after Idle "The IEEE 802.11 protocol sets a DIFS counter once the communication channel becomes idle. Before sending an RTS packet, a node will wait a random period of time between 0 and DIFS", as the authors of [10, RAP] said. To prioritize this process, the RAP sets the DIFS parameter based on the packet priority level:

$$DIFS = Base\_DIFS \times Priority\_Level$$

So the packets fall in a higher priority range level (smaller priority value) would wait shorter on average.

Backoff Increase Function When a collision is sensed in MAC layer, the IEEE 802.11 protocol would double the backoff window, CW, to extend the waiting period of a packet before it is forwarded again. The RAP modified the window backoff algorithm to differentiate the packets with different priorities, as:

$$CW = CW \times (2 + (Priority\_Level - 1)/Max\_Priority\_Level)$$

where $Max\_Priority\_Level$ is the highest priority level. The backoff window of a packet with lower priority increases faster than that of one with higher priority.

Chapter 5

JiTS: Just-in-Time Scheduling Mechanism

In this chapter, we describe the proposed Just-in-Time scheduling (JiTS) framework. JiTS takes an integrated approach to real-time packet delivery: it considers packet scheduling, queueing as well as routing. JiTS can interoperate with existing routing protocols for sensor networks. Moreover, it does not require any support from or changes to the lower layer protocols. We first overview the JiTS approach and then discuss the implementation in detail.

5.1 Just in Time Scheduling

5.1.1 Intuition and Motivation

In a light taffic environment, data packets are forwarded along the path from the source to sink quickly. The main contribution of the end-to-end tavelling time is the processing and lower layer transmission delay. In the light traffic environment, any scheduling algorithm may not help the end-to-end performance much. For the real-time data dissemination, even routing protocol without scheduling may meet the end-to-end real-time constraints at most time.

In heavy traffic environments, large queueing delays may be experienced at intermediate nodes before they are forwarded to their next hop. This situation may occur at several intermediate nodes before the packet reaches its destination. The design of the real-time

scheduling algorithm should not ignore this part of time contribution. Unfortunately, existing protocols such as SPEED[11, 12], RAP[10] and MMSPEED[13] all do not account for this component of the delay directly.

In traditional multi-hop communication systems, any packet in a forwarding queue would be forwarded as soon as there is no other packet with higher priority waiting to be forwarded. Traditional scheduling algorithms do not *delay the packets*. Given that packet transmission indirectly affects other nodes which may have urgent packets (by making the medium busy and delaying their transmission), we question whether immediately transmitting packets at each intermediate node results in a *globally optimal behavior*. To answer this question, we designed the *Just-in-Time Scheduling* (JiTS).

5.1.2 JiTS: Basic Algorithms

JiTS is the primary contribution of this thesis. The first distinguishing feature of JiTS is that it considers all components of delay, including queueing delay at each forwarding node. In addition, JiTS delays data packet transmission during forwarding for a duration that correlates with their remaining deadline and distance to the destination. Intuitively, this helps in heavy-traffic communication environment by making sure that priority inversion does not occur due to a node with only low priority packets sending and preventing a node with high priority packets from doing so. Moreover, delaying the data packets before reaching the sink also helps the data aggregation/fusion and therefore energy efficiency; we do not explore this effect in this paper.

Before a data packet reaches the sink, the end-to-end transmission and processing delay cannot be obtained. Therefore, we use previous measurements of delay to estimate the overall delay: we call this estimate the end-to-end *Estimate of Transmission Delay* (EETD). As discussed in section 3.1.2, the one hop estimate is called ETD. Summing the ETD's of a data packet hop by hop during its forwarding can lead to inaccurate estimates

since one hop ETD can fluctuate significantly. Therefore, we use the following function to decide the EETD:

$$EETD = ETD \cdot \frac{E2E\ distance}{One\ hop\ distance} \tag{5.1}$$

where the distance can be measured in different ways.

Different JiTS scheduling policies can be developed based on the allocation of the available slack time among the different hops. The target transmission times are either set by the source or computed at intermediate hops based on a known algorithm. In the base JiTS algorithm, the target transmission time is set to be equal at all hops and is determined as follows:

$$Target\ Delay = \frac{deadline - EETD}{distance(X, sink)} \cdot \alpha \tag{5.2}$$

where the α is a constant "safety" factor for insurance that the real-time deadline would be met. For example, setting to be 0.7, would target delaying the packet 70% of the available slack time, leaving the remaining time as a safety margin.

As we can see, the Target Delay of any in-queue packet determines its priority. The time a packet is delayed in the queue can be used as the key to a priority queue that holds the packets to be transmitted. The end-to-end transmission and processing delay is considered along with the queueing delay, by taking into accout the end-to-end deadline, distance and EETD.

We consider static vs. dynamic versions of the protocols depending on whether the target transmission times are set by the source and followed by intermediate nodes (static), or whether they are computed/adjusted at intermediate nodes (dynamic).

Static Just-in-Time Scheduling In static JiTS, the target delay is set with the values of parameters at the data source. In the equation5.2, the end-to-end deadline is fixed at the data source; the EETD is measured with the ETD of forwarding node and the distance from source to sink (X is the data source). So even we call it static, the different ETD's of

forwarding nodes would make the target delay at each node different.

Dynamic Just-in-Time Scheduling In dynamic JiTS, the target delay is re-set at each forwarding node with the local value of parameters. In Equation5.2, the end-to-end deadline of a packet at some forwarding node is the *remaining slack time*, measured by $E2E\ Deadline - Elapsed\ Time$. The EETD is decided by the one-hop ETD of the forwarding node and the distance from it to the sink, not the distance from source to sink. So the dynamic JiTS is able to continuously refine the priority of the packet.

Non-linear Just-in-Time Scheduling It is also possible to allocate the available slack time nonuniformly among the intermediate hops along the path to the sink. For example, we may desire to provide the packets with additional time as it gets closer to the sink. The intuition is that in a gathering application, the contention is higher as the packet moves closer to the sink. Different policies can be developed to break down the available time. We explore the following policy:

$$Target\ Delay = \frac{E2E\ Deadline - EETD}{2^{\frac{Remaining\ Distance}{One\ hop\ Distance}}} \cdot \alpha \tag{5.3}$$

More generally, we may want to allocate the slack time proportionately to the degree of contention along the path. Such a heuristic may be developed by passing the contention information along with the routing advertisement and allocating the available slack time accordingly. Finally, one may decide to favor aggregation by delaying packets more closer to the source where the data is more correlated. We do not explore this policy in this thesis; however, JiTS provides the flexibility of this nonlinear slack allocation.

5.2 JiTS Implementation

JiTS does not ignore the queueing delay. It considers both the transmission delay and the queueing delay by doing a set of very simple scheduling decisions. The basic JiTS scheduling algorithm has been shown in section5.1. But JiTS is more than that.

Although the term *JiTS* stands for Just-in-Time Scheduling, it is not only a scheduling algorithm. It involves the architecture design of the whole system. The typical architecture of a system that JiTS works on is shown in figure5.1. The JiTS scheduler resides above

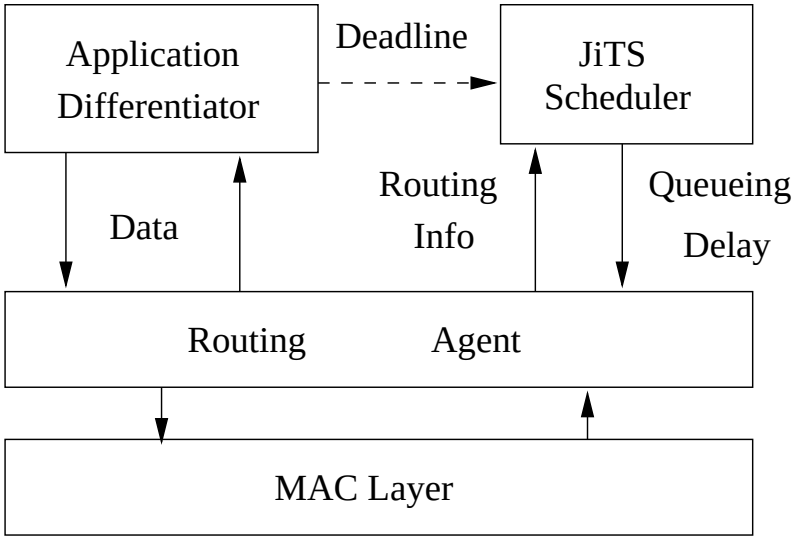


FIG. 5.1. The architecture of a system centered at JiTS

(or within) the routing layer. It uses routing level information such as the end-to-end distance in making its scheduling decisions. For any real-time applications based on sensor networks, the end-to-end real-time deadline is assumed to be included on the data packet itself. Figure5.1 shows an example of how this information is collected. While, in this figure, the MAC layer is shown, the JiTS scheduler and the MAC layer protocol are not aware of each other.

5.2.1 JiTS for different routing protocols

JiTS can be adapted to work with virtually any underlying routing protocol. However, the JiTS algorithm may need to be adapted to consider the cost metric used by the routing algorithm. For example, in a system based on the shortest path routing (SP), the distance parameters used by JiTS scheduler is measured in number of hops. The corresponding functions are:

$$EETD = ETD \cdot E2E\ hops$$

$$Target\ Delay = \frac{deadline - EETD}{h} \cdot \alpha$$

$$Target\ Delay = \frac{E2E\ Deadline - EETD}{2^h} \cdot \alpha$$

where h stands for end-to-end number of hops. For the geometric routing, the values of distance parameters used in JiTS Scheduler would be the Euclidean distance.

In summary, the following information is needed to schedule packets in JiTS:

- *End-to-end deadline information*: this information is provided by the application in the data packet as required by any real-time data dissemination application. For those applications where the header of data packet does not include this information, an alternative way for JiTS to obtain the end-to-end deadline information is needed.

- *End-to-end distance information*: this information is obtained from the routing protocol. For example, this information is maintained in the routing tables of traditional distance vector based or link-state based routing protocols to keep track of the cost of the path. Furthermore, in source routed protocols such as DSR, this information can be directly computed from the packet header which includes the full path to the destination. Finally, in geographic routing, Eucleadian distance measured as the distance from the current node to the destination can be used as the distance metric.

The output of JiTS scheduler is the queueing delay, which is used by the routing pro-

tocol to decide how long to delay an incoming data packet before attempting to forward it (by passing it to the MAC layer). MAC layer prioritization is not needed by the JiTS design since the packets are sent when their Just-in-time local deadline is reached – they should all be of roughly equal priority. Not requiring changes to the MAC layer is a desirable feature of JiTS relative to RAP.

5.2.2 Priority Queue

A single priority queue is used in the routing layer for forwarding packets. A single priority queue can better track the priority of data packets than the approach of several FIFO queues with different priority levels. The processing time of queueing operations is much shorter than the queueing delay, so this overhead does not affect the performance much as it does in RAP design. The priority of each data packet is the queueing delay decided by the JiTS scheduler. This priority does differentiate not only the requested velocity of each data packet, but also their possible queueing status.

When the priority queue is full, the data packet at the head of the queue is forwarded at once no matter how long it needs to be queued due the JiTS scheduling. As we can see, this is a best effort forwarding policy which may cause the most urgent data packet to be dropped. However, the benefit of this solution is that the packet at the queue head has a higher local priority and is less likely to be dropped by lower layer communication later.

5.3 Properties of JiTS

In summary, the following are the design features of the JiTS framework:

- Ability to interoperate with different routing protocols: unlike the SPEED[11] or RAP[10] framework which are specific to geographical routing, JiTS is not limited to a specific routing protocol. Instead, it can operate directly with any hop-based cost-metric protocol and can be easily adapted to work with Geometric routing protocols.

This flexibility is demonstrated via simulation later in this thesis.

- Soft Real-time: JiTS maintains an uniform delivery speed of data packets, meeting the deadline of most data traffic with best effort. Packets that pass their deadline are not dropped. While its possible to better support hard real-time in this framework (for example, by increasing the safety margin, and immediately dropping packets that are late), we do not pursue such extensions.

- No MAC layer support required: Unlike the SPEED or RAP, JiTS does not require MAC layer support for prioritized scheduling (as with RAP) or for tracking delay (as with SPEED). This makes JiTS readily deployable on existing infrastructure.

- QoS routing: JiTS integrates the tansmission delay with the queueing delay, considering both the lower layer communication cost and that of higher layers and differentiating the data flows with different real-time constraints.

- Ability to withstand high load and hotspotting: JiTS uses the queueing mechanism to delay any data flows to restrict contention to occur among only the most urgent traffic. This allows JiTS to gracefully accomodate higher traffic levels than RAP or SPEED.

- Data Fusion: JiTS tries to delay any incoming data traffic which gives more possibility of the data aggregation operations. Since the data aggregation is a primary data operation during the data forwarding for most applications, JiTS fits better than the other approaches which attempt to send packets without delay.

Chapter 6

Implementation and Experimental Evaluation

We implemented JiTS (Static, Dynamic and Non-Linear) with both the Shortest Path routing and Greedy Geographic Forwarding in the Network Simulator (NS2, version 2.27)[59]. We also implemented the RAP Velocity Monotonic Scheduling (VMS) with GF, including the specialized MAC support required by it on NS2 per the specification. We also implemented the SPEED protocol (however, the measured performance for SPEED does not validate well against the results reported in the original paper). Since GF has been shown to significantly outperform traditional routing protocols such as DSR[14] and deadline-based scheduling, in the context of sensor network data dissemination, we restrict the routing comparison to GF and SP, and the scheduling comparison to VMS and JiTS.

6.0.1 Simulation details

Mac layer protocol	IEEE 802.11 with prioritizing extension
Transmission Radio Range	250 m
Bandwidth	2Mbps
Data Packet Size	32B
Data Rate	2 packtes/sec
Simulation Area	$1000 \times 1000m^2$
Number of Sensor nodes	100
Effective Simulation Time	120s

Table 6.1. Simulation Parameters

Table6.1 shows the simulation parameters we use; unless otherwise indicated these parameters are used in the studies. We use both the grid and random deployment to simulate our algorithm. In grid deployment, we divide the covered simulation area into a 10×10 grid. One of the 100 sensor nodes is placed at the center of each the grid tiles. The sink is placed on the northwest corner of the network. Nodes publish data at the rate of 2 packets per second in order to simulate a fairly high load traffic scenario. In random deployment, the 100 nodes are randomly placed in the simulation area while the sink is placed roughly at the center of the area.

First, we compared JiTS with VMS both using the same routing protocol (GF); recall that GF was used in the original RAP scheme[10]. Later, we show that SP significantly outperforms GF for JiTS. Since we consider soft real-time applications, a change we made to the RAP mechanism is that each node tries to forward all incoming data packets, no matter if the deadline is already missed or not. In the original implementation of RAP, the packets missing the deadline would be dropped. Since JiTS does not require any MAC layer information, we use the original IEEE 802.11 as our MAC layer protocol.

We considered the issue of what the JiTS safety margin parameter α should be set to. If α is too high, packet delay variability can cause deadlines to be missed since most of the slack time is taken up by intentional JiTS delay and unexpected delays cause a packet to miss the deadline. Conversely, if the α is too low, packets are conservatively sent quickly towards the sink, possibly overflowing buffers around it. Experimentally, we observed that a safety margin parameter of 0.7 works well across different deadlines. Thus, 30% of the deadline budget is set aside to account for inaccuracies in ETD estimates and/or unexpected transmission or queuing delays.

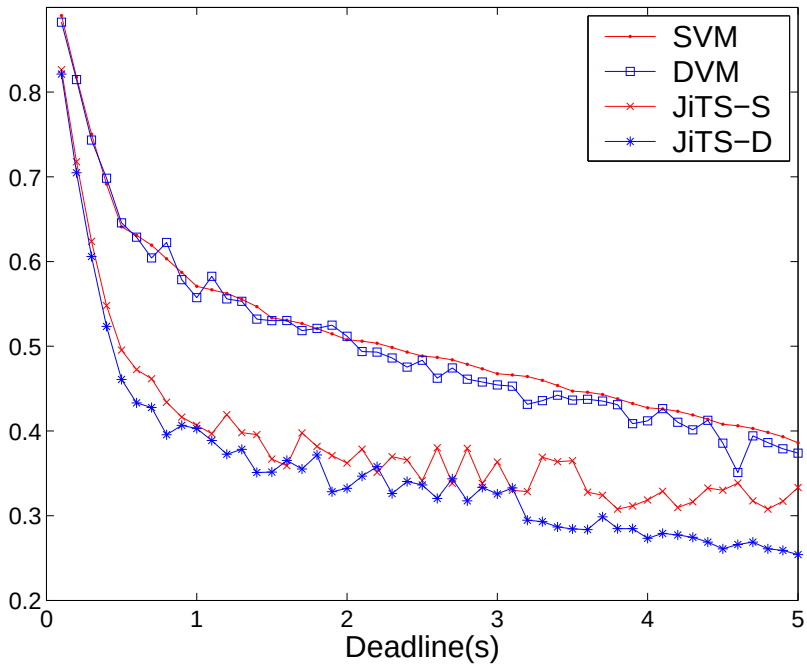


FIG. 6.1. JiTS(GF) vs VMS Miss Ratio

6.1 Performance Evaluation

6.1.1 JiTS vs Velocity Monotonic Scheduling

The first experiment studies the performance of JiTS scheduling for sensor networks relative to RAP. Figure 6.1 and figure 6.2 show that for different deadline requirement, the miss ratios and drop ratios of JiTS Static and Dynamic are much lower than those of DVM and SVM for across all the considered deadline range. Dynamic JiTS outperforms static JiTS in terms of the miss ratio.

Figure6.3 shows the average delay of JiTS and that of RAP to illustrate the difference between the two scheduling approaches. The average delay of JiTS grows linearly with the deadline as the intermediate nodes delay packets proportionately to the deadline. Note that dynamic JiTS manages to keep the average delay around the value of $0.7 \cdot Deadline$, while the static JiTS has slightly higher average delay. Since RAP does not delay packets, its

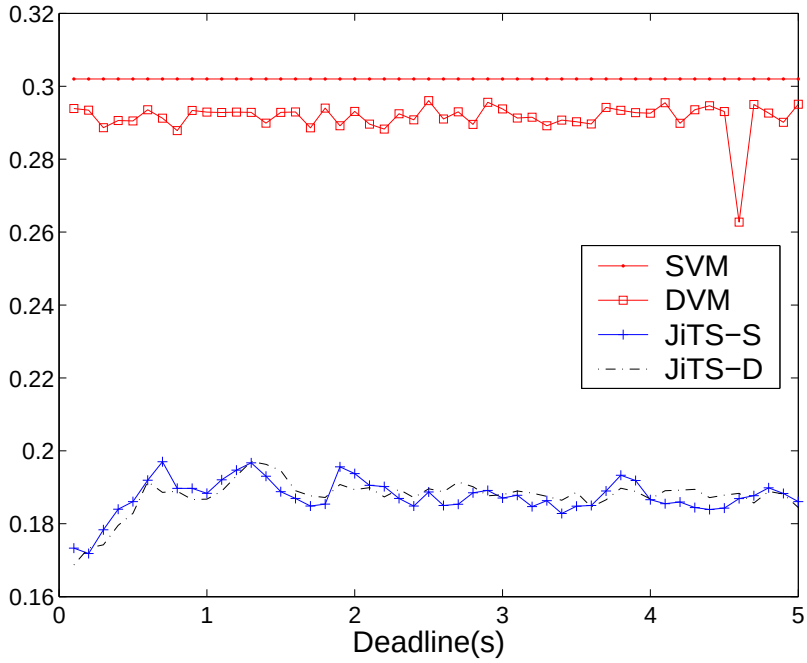


FIG. 6.2. JiTS(GF) vs VMS Drop Ratio

delay depends only on the data generation pattern and does not change with the deadline. Since VMS uses multiple FIFO queues as its priority queue, packet starvation commonly occurs and the maximum packet delay suffered by RAP is much worse than that of JiTS (by a factor of 2 to 3). Since the performance of SVM and DVM is nearly the same, and since the authors of RAP observed SVM to be superior to DVM [10], we will only show results with SVM for the remainder of this chapter.

6.1.2 Effect of Routing Protocol: SP vs GF

In this second experiment, we compare the performance of JiTS under GF with JiTS with Shortest Path (SP) routing. In addition, in this experiment, we include the performance of the nonlinear JiTS scheduling algorithm (JiTS-NL); recall that JiTS-NL delays the packet non-uniformly, allowing more slack time as it gets closer to the sink. Figure 6.4 and Figure 6.5 show the miss ratio and drop ratio respectively. Clearly, JiTS performs con-

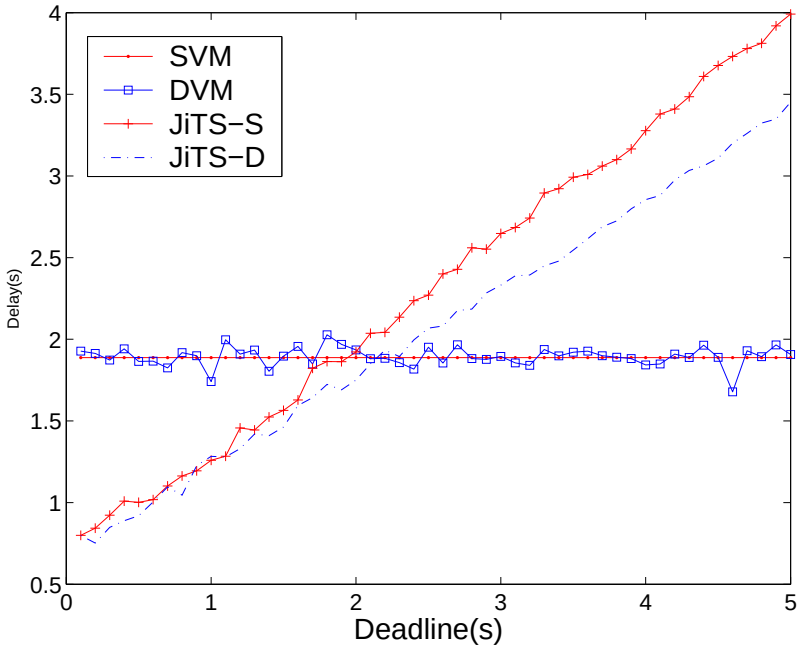


FIG. 6.3. JiTS(GF) vs VMS Average Delay

siderably better with SP than with GF. In general, dynamic JiTS performs better than static JiTS for both routing protocols. Furthermore, JiTS-NL provides significant improvement over both static and dynamic JiTS; the improvement is most pronounced under tight deadlines. The maximum and average delay of JiTS with GF is higher than that with SP for the same deadlines, results shown in figure 6.6. This increased delay can be explained by the fact that GF uses longer paths than SP in terms of the number of hops. For the remainder of this paper, we use SP as the routing protocol for JiTS.

6.1.3 Performance under Bursty Traffic

In this study, we evaluate the performance of JiTS vs. RAP under bursty traffic conditions. Each node publishes alternately publishes packets at the pre-set data rate for 5 seconds then stops publishing for the second 5. Figure6.7 shows the miss ratios and drop ratio of JiTS and SVM under this bursty traffic with end-to-end deadline from 0.1 second

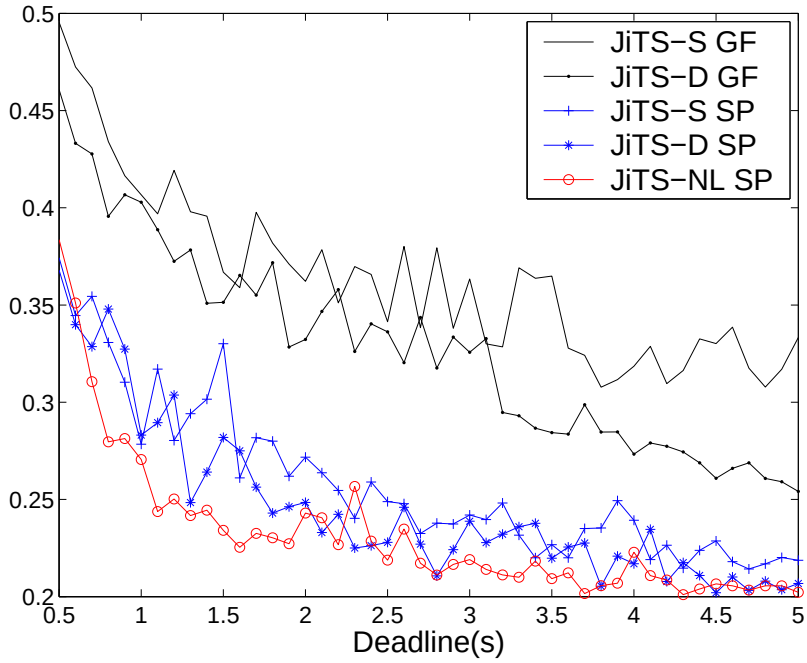


FIG. 6.4. JiTS SP vs GF Miss Ratio

to 3.0 seconds. From the figure we can see that the miss ratio of dynamic JiTS is much lower than that of SVM with the bursty traffic, because JiTS can tolerate the traffic burst by delaying some packets, and taking advantage of the idle period. On the other hand, SVM cannot make use of the traffic behavior since it does not delay packets. The decrease in the drop ratio shows that JiTS also deliveries more packets as the deadline constraints are relaxed (figure6.8).

6.1.4 Performance under Random Deployment

JiTS and VMS were also evaluated using a random deployment scenario where the 100 nodes were randomly placed within the simulation area. Figures6.9 show three random deployments of 100 sensor nodes in a $1000 \times 1000m^2$ area. Each result represents the average of several experiments with different seeds. We varied the deadline requirements from 0.5 to 2.0 seconds in steps of 0.5 seconds. Figure6.10 and figure 6.11 show the miss

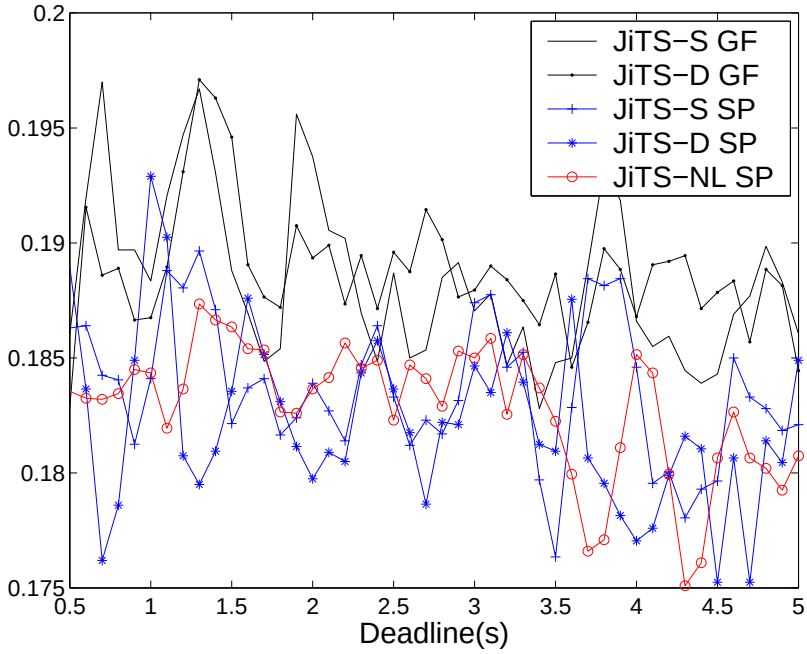


FIG. 6.5. JiTS SP vs GF Drop Ratio

ratios and drop ratios for the different algorithms. The simulations show that both JiTS and VMS perform much better in a random scenarios than they did in the grid scenarios possibly because the location of the sink is central to the simulation area, making the average sensor distance to the sink smaller. Again JiTS provides superior performance to VMS. For the VMS, the drop ratios do not decrease as the deadline grows since it prioritizes but does not delay packets. The drop ratio becomes the lower bound of the miss ratio. JiTS shows more reactivity since both the drop ratio and miss ratio keep decreasing as the deadline requirement is relaxed.

6.1.5 Performance with Multiple Deadline Data

We simulated the presence of two data types being generated by the sensors with two different deadline constraints (deadline of level 1 data is half that of level 2 data), in bursts. Ideally, the scheduling algorithm would allow both data types to meet their deadlines effec-

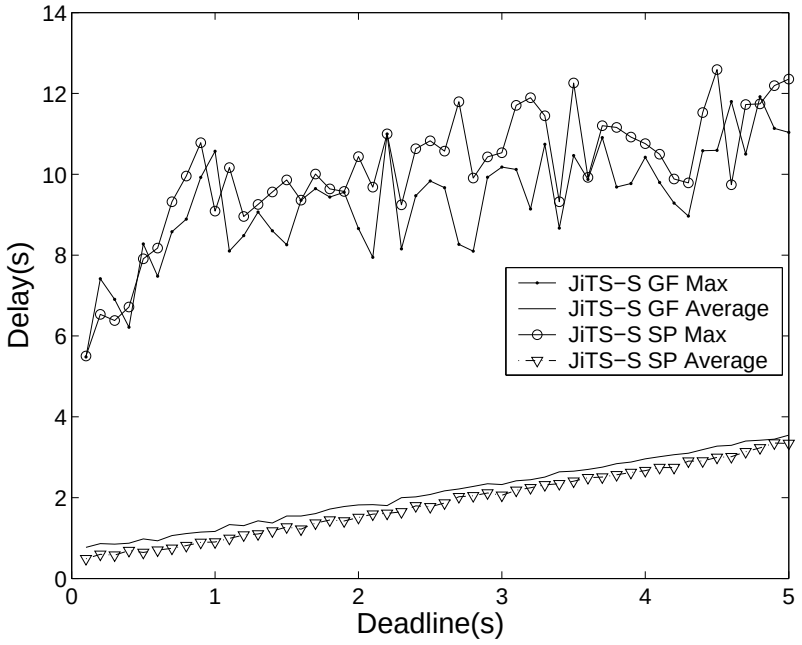


FIG. 6.6. Delay of JiTS-S: GF vs SP

tively. Figure6.12 and figure 6.13 show the miss ratio and drop ratio of VMS, JiTS Dynamic and Non-Linear. Under strict deadlines, level 2 traffic receives better performance because the network was often unable to satisfy the aggressive level 1 deadlines. Once the deadlines grow beyond a certain level, JiTS is able to provide similar performance to the two traffic types, despite their different deadline requirements. The same is not true of RAP, where level 2 traffic continues to receive better performance than level 1 traffic.

6.1.6 Comparison with SPEED

We also built simulation models for the SPEED framework[11] within the NS-2 simulator. Unfortunately, the simulation results we obtain does not match the performance demonstrated in the original SPEED papers [11, 12]. We implemented the full specification of SPEED, SPEED-T (Minimal one hop delay first), and SPEED-S (maximal one hop progress speed first), simulating them in the exactly same scenarios specified in

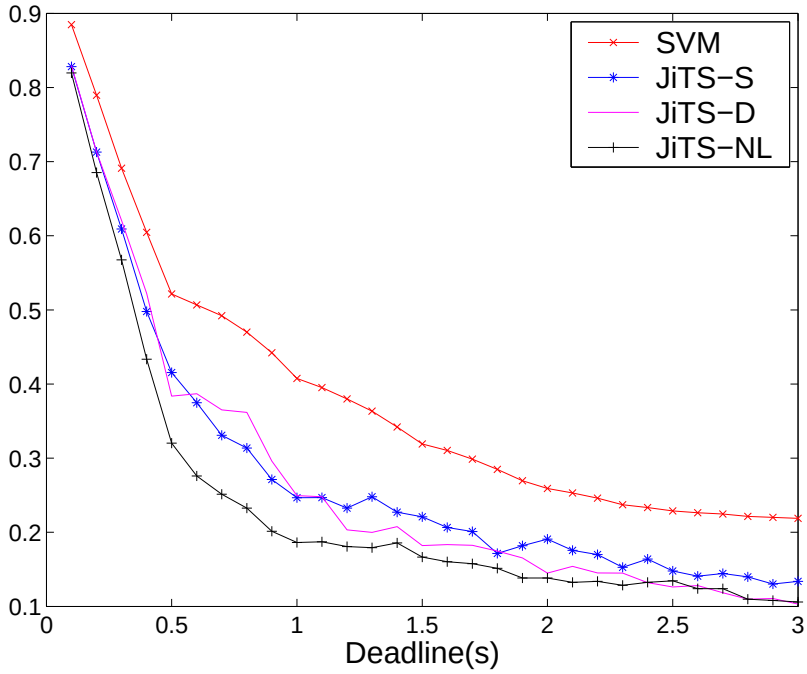


FIG. 6.7. JiTS vs SVM with bursty traffic Miss Ratio

[11, SPEED]. We then compared the simulation results againt the shortest-path routing and greedy forwarding routing in a void-free deployment. Figure 6.14 shows the performance of SPEED against greedy forwarding as per the original paper specification of SPEED(figure is quoted from the original SPEED paper[11]).

For both evaluation metrics, SPEED performs better than greedy forwarding. When the data rate of the congestion-introducing flow is lower than or equal to 50 packets per second, the end-to-end delays of SPEED and GF are both lower than 100 milliseconds, the miss ratios of both are lower than 5%. If the data rate of that flow is equal to or higher than 60 packets per seconds, the end-to-end delays of both increase higher than 140 milliseconds, and miss ratios are higher than 15%. The performance of SPEED is much better than GF for the data rate of the interfering flow is high.

Figure 6.15 and figure 6.16 show the simulation results of our implementation of SPEED in NS2. For the congestion-introducing flow with lower data rate, the SPEED

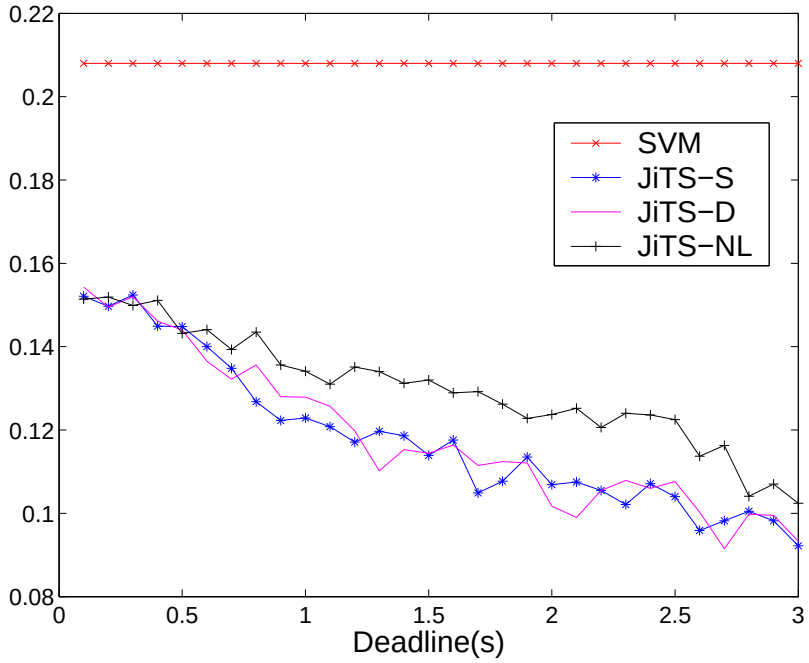


FIG. 6.8. JiTS vs SVM with bursty traffic Drop Ratio

and GF perform similarly as the results of [11, SPEED]. However, the end-to-end delays never increase higher than 90 milliseconds. In term of end-to-end delay, SPEED is better than both the greedy forwarding and shortest path routing in most cases, especially when the data rate of congestion-introducing flow is low. However, the miss ratio of SPEED remains higher than both greedy forwarding and shortest path routing in all cases. It is very high even when the data rate of congestion-introducing flow is low. The miss ratios of greedy forwarding and shortest path routing are not as high as reported in [11, SPEED].

Figure 6.17 shows the drop ratio of our implementation of SPEED against both the greedy forwarding and shortest path routing. The drop ratios of our implementation of SPEED is much higher than shortest path routing. It is also a little bit higher than greedy forwarding. As we can see, the high miss ratio of our implementation of SPEED mostly comes from its high drop ratio. To compare the greedy forwarding and shortest path routing, as expected, the shortest path routing performs better than greedy forwarding. Since

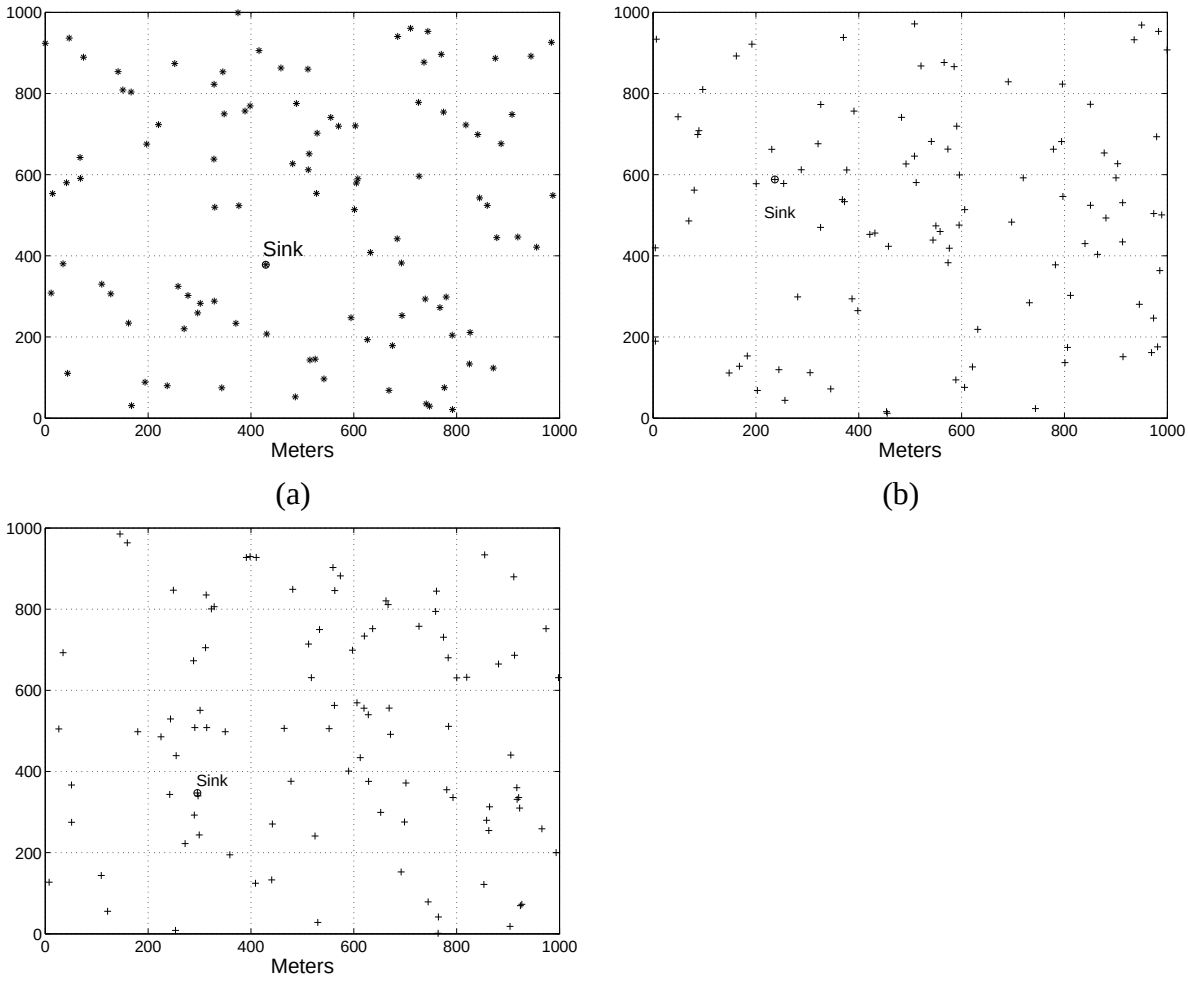


FIG. 6.9. Random Deployments of 100 sensor nodes

the drop ratio of shortest path routing is much lower than the greedy forwarding, it seems that more data packets with longer end-to-end delay are received by shortest path routing, which leads to the higher average end-to-end delay shown as seen in Figure 6.15.

Possible Reasons for the Difference in observed Results The possible reasons why we can not repeat the originial simulation results of SPEED are:

- Simulation Model Discrepencies: The design of SPEED involves MAC and routing protocols. It introduces some extension of IEEE 802.11 MAC protocol, which

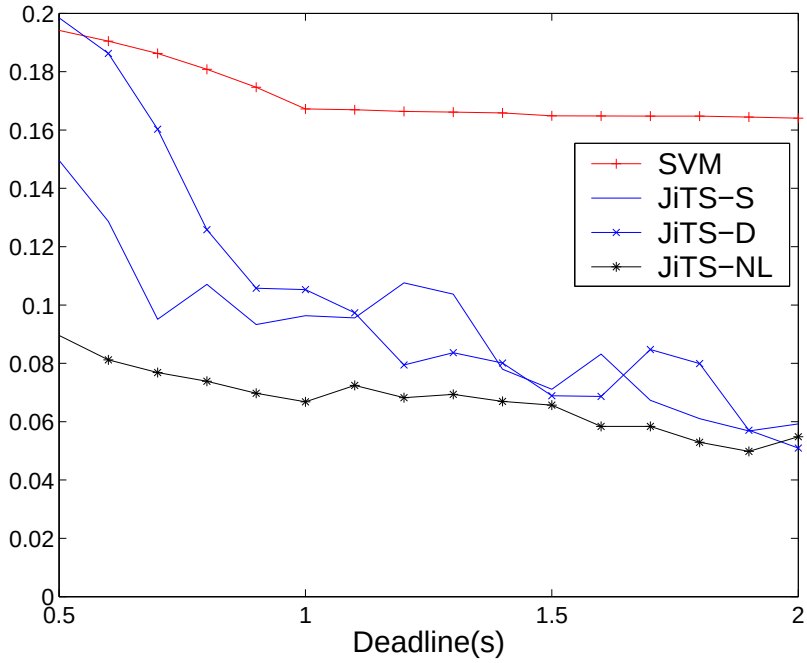


FIG. 6.10. JiTS vs SVM with Random Scenario Miss Ratio

involves the implementation of not only the MAC and routing protocols in NS2, but also the physical layer implementation. I am not sure if my implementation of SPEED matches that of the original authors since SPEED requires changes at several subsystems of the network protocol.

- Mimatch in parameter settings: In the design of SPEED, several parameters are very important. The design of SPEED [11, 12] does not specify how to select good values for all these parameters. In personal correspondence with the first author, he explained that those parameters have to be selected manually and vary with the scenario.

- Differences between simulators: the original implementation of SPEED is based on another simulator, GloMoSim [60], developed by UCLA. Our implementaion is based on NS2. The two simulators are quite different in many areas, especially the lower level models.

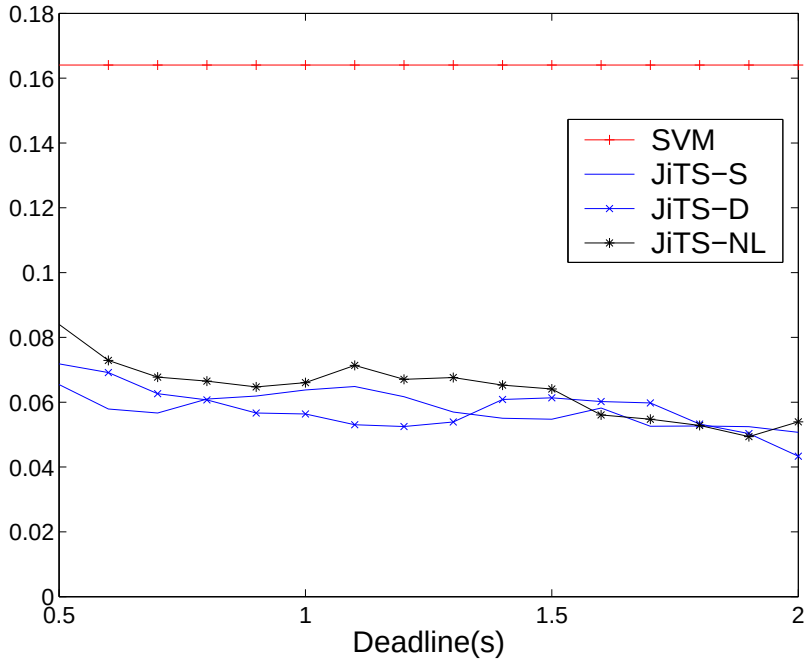


FIG. 6.11. JiTS vs SVM with Random Scenario Drop Ratio

Overall, our experiences with SPEED show that it performs extremely poorly at high loads because its backpressure mechanism is not suited to the situations where alternative paths are also congested. In those situations backpressure ends up increasing the load on the network by routing packets through unecessarily long paths. We believe that our comparison is fair because all the algorithms are implemented in the same environment (thus removing any differences that occur due to the different simulators). Because of the overall poor performance under high load, we do not compare JiTS with SPEED in detail.

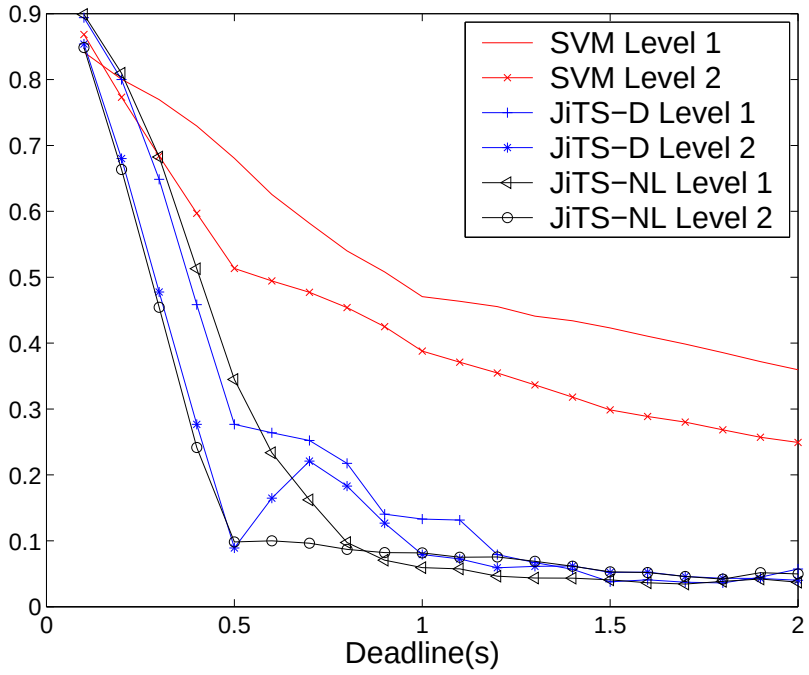


FIG. 6.12. JiTS vs SVM: 2 levels of deadlines Miss Ratio

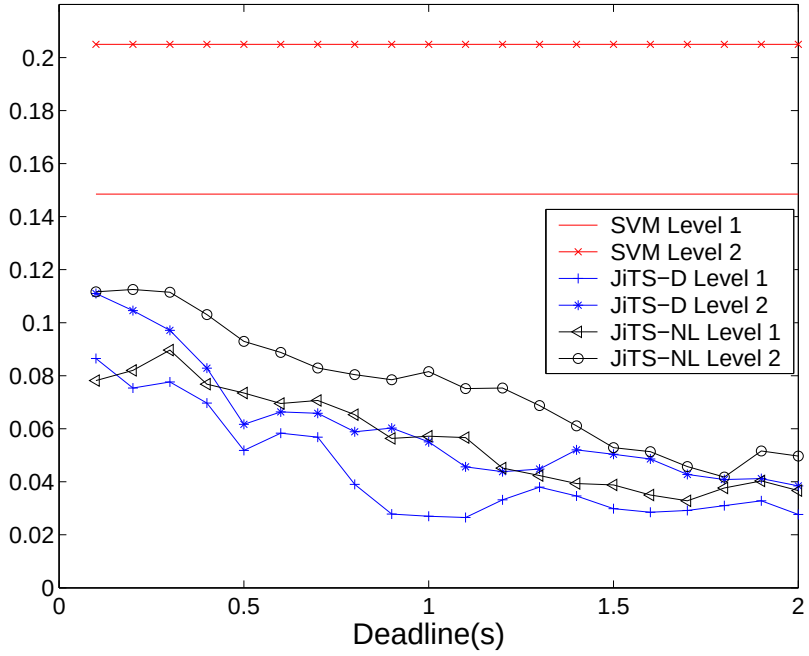
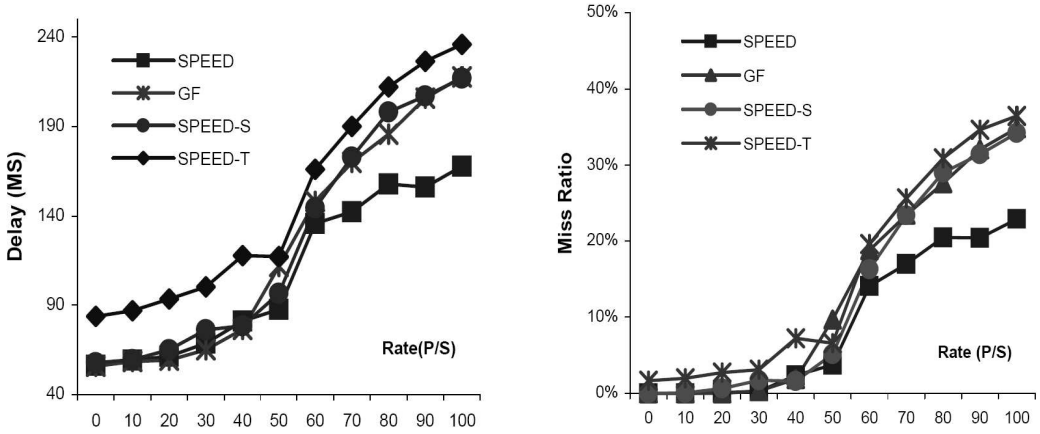


FIG. 6.13. JiTS vs SVM: 2 levels of deadlines Drop Ratio



End-to-end Delay End-to-End Miss Ratio

FIG. 6.14. Original simulation results of SPEED

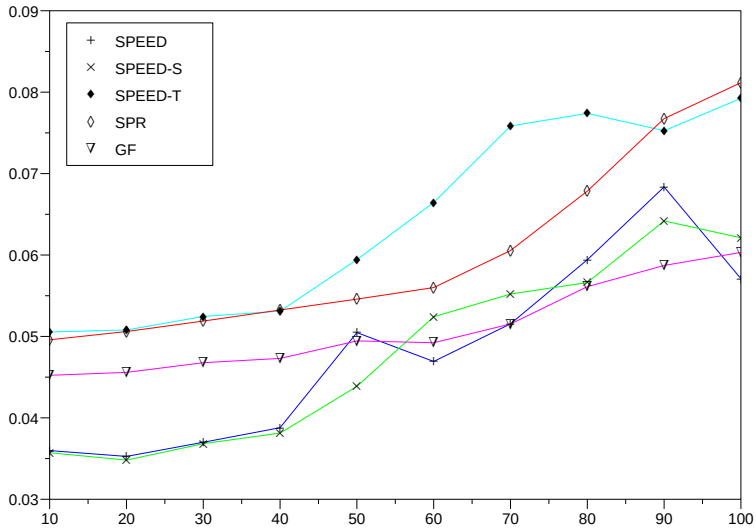


FIG. 6.15. Our implementation of SPEED in NS2: end-to-end delay

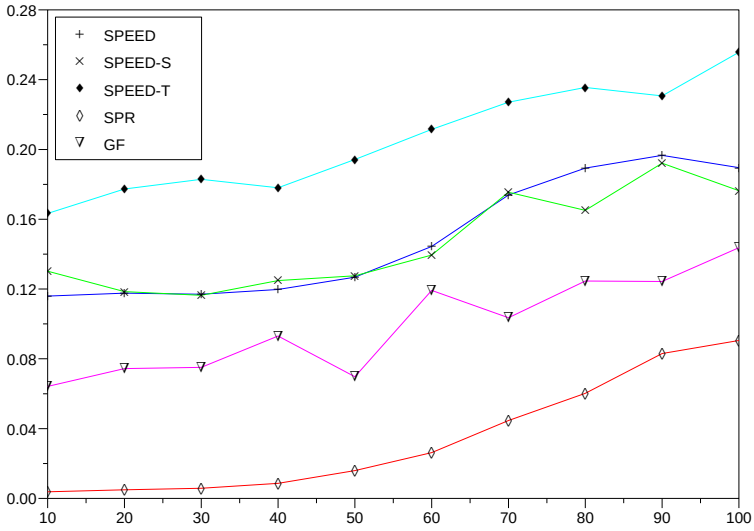


FIG. 6.16. Our implementation of SPEED in NS2: miss ratio

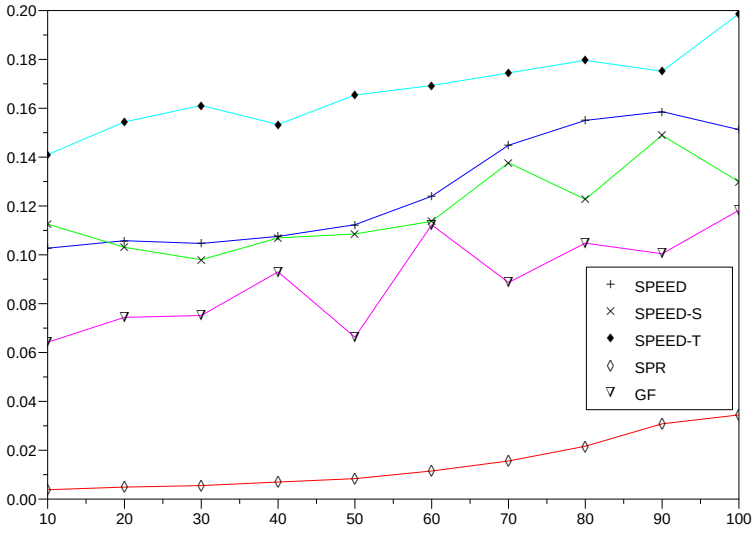


FIG. 6.17. Our implementation of SPEED in NS2: drop ratio

Chapter 7

Conclusions and Future Work

Real-time data dissemination is a service of great interest to many sensor network applications. The thesis proposed and evaluated the Just-in-time scheduling mechanism for real-time sensor network applications. JiTS offers significant advantages over existing real-time sensor data dissemination schemes. It accomplishes real-time support by delaying packets a fraction of their slack time at each hop. As a result, it is better able to tolerate bursts than schemes that simply prioritize packet transmission.

We also explored the effect of routing on real-time scheduling success and showed that Geographical Forwarding can lead to suboptimal operation. JiTS can operate with simple routing protocols easily and outperforms RAP in both the miss ratio and overall delay. The thesis explored criteria for allocating the available slack time among the different nodes and showed that nonlinear distribution of the slack time, with more time assessed to hops closer to the sink results in better performance than linear distribution of the slack time in the gathering scenarios that we studied. JiTS is a network layer solution and does not require changes to lower level protocols making it easier to deploy and independent of the underlying sensor network hardware capabilities.

Using simulation, we found the drop ratio is the lower bound of the miss ratio of real-time communication. If the drop ratio is decreased, given a reasonable end-to-end deadline, the miss ratio of these real-time applications should also be decreased. Mostly

the packets are dropped due to congestion as the network capacity is exceeded. We would like to study the effect of queueing in the lower layers, which would involve congestion control of network traffic[55]. By analyzing the traffic behavior, we can easily realize that when a packet approaching the sink, intermediate nodes have to forward more and more packets, which means the traffic would increase as packet approaches the sink. It makes sense that delay the packet longer when it is closer to the sink. How to assign the queuing delay according to its distance to the hops requires more careful study. This is especially true when we consider that aggregation would favor packets being delayed closer to the source. Coming up with JiTS policies that are sensitive to these requirements is a topic of future research.

JiTS focuses on scheduling packets under heavy traffic situations. We did not examine the problem of scheduling under a light load. If the traffic through the current node is not heavy and queuing time is small real-time scheduling is not needed and in fact may harm performance. If such a situation can be detected, JiTS can be disabled and packets forwarded normally. For example, an idle detection mechanism may be employed such that if an idle period passes without packet transmission, the top of the queue is sent immediately.

A routing protocol for special real-time applications[11, 12] may be used by JiTS so as to enhance the performance. Since the end-to-end distance in a geometric routing is difficult to decide before the data packet reaches the sink, developing a more accurate algorithm to estimate this parameter for JiTS is also needed. This need is especially important since geometric routing is gaining momentum as a preferred routing approach in large scale sensor networks.

References

[1] Smart Dust. http://www-bsac.eecs.berkeley.edu/archive/users/warneke-brett/SmartDust/. DARPA project.

[2] Parc DARPA CoSense Project. http://www2.parc.com/spl/projects/cosense/.

[3] DARPA SENSIT Project: Self-organizing sensor network. http://www.eng.auburn.edu/users/lim/sensit.html.

[4] CENS: center for embedded networks sensing. http://research.cens.ucla.edu/.

[5] Nest: Network of embedded systems. http://www.cs.virginia.edu/ control/nest.html. NEST project, VigilNet, University of Virginia.

[6] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor networks. In *Computer Magzine*, pages 41–49. IEEE Computer Society, August 2004.

[7] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263–270, Seattle, Washington, USA, August 1999. ACM.

[8] Deborah Estrin, David Culler, Kris Pister, and Gaurav Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, January–March 2002.

[9] John Stankovic, Tarek Abdelzaher, Chenyang Lu, Lui Sha, and Jennifer Hou. Real-time communication and coordination in embedded sensor networks. In *Proceedings of the IEEE*, volume 91, pages 1002–1022, July 2003.

[10] Chenyang Lu, Brian M. Blum, Tarek F. Abdelzaher, John A. Stankovic, and Tian He. RAP: A real-time communication architecture for large-scale wireless sensor networks. *Real-Time and Embedded Technology and Applications Symposium, 2002*, 24-27 Spet. 2002.

[11] Tian He, John A Stankovic, Chenyang Lu, and Tarek Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. *International Conference on Distributed Computing Systems(ICDCS 2003)*, May 2003.

[12] Tian He, John A. Stankovic, Chenyang Lu, and Tarek F. Abdelzaher. A spatiotemporal protocol for wireless sensor network. *IEEE Transactions on Parallel and Distributed Systems*, To appear.

[13] Emad Felemban, Chang-Gun Lee, Eylem Ekici, Ryan Boder, and Serdar Vural. Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks. *IEEE INFOCOM 2005*.

[14] David B. Johnson, David A. Maltz, and Josh Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In Charles E. Perkins, editor, *Ad Hoc Networking*, pages 139–172. Addison-Wesley, 2001.

[15] B. Karp and Kung. H. T. GPSR: Greedy perimeter stateless routing for wireless networks. *Proc. 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, 2000.

[16] Prosenjit Bose, Pat Morin, Ivan Stojmenovi, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of 3rd ACM Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications DIAL M99*, August 1999.

[17] Guoliang Xing, Chenyang Lu, Robert Pless, and Qingfeng Huang. On greedy geographic routing algorithms in sensing-covered networks. *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'04)*, May 2004.

[18] vangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proceedings of 11th Canadian Conference on Computational Geometry*, 1997.

[19] G.G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, Information Science Institute, March 1987.

[20] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worstcase optimal and averagecase efficient geometric adhoc routing. In *MobiHoc 2003*, June 2003,.

[21] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MobiCom 2003*, Sept. 2003.

[22] Antonio Caruso1, Stefano Chessa1, Swades De1, and Alessandro Urpi. GPS free coordinate assignment and routing in wireless sensor networks. In *INFOCOM 2005*, March 2005.

[23] S. Corson. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. RFC 2501, January 1999.

[24] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[25] Sameer Tilak, Nael B. Abu-Ghazaleh, and Wendi Heinzelman. A taxonomy of sensor network communication models. *Mobile Computing and Communication Review*, 6(2), April 2002.

[26] Sameer Tilak. Sensor network communication models. Master's thesis, SUNY Binghamton, 2002.

[27] B. O Hara and A. Petrick. *IEEE 802.11 Handbook - a Designer s Companion*. IEEE Press, 1999.

[28] F. A.Tobagi and L. Klienrock. Packet switching in radio channels: Part ii the hidden terminal problem in carrier sense multiple access and the busy tone solution. *EEE Transaction on Communication*, I(COM-23):14171433, 1975.

[29] Phil Karn. MACA - A New Channel Access Method for Packet Radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134–140, London, Ontario, Canada, September 22 1990.

[30] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A Media Access Protocol for Wireless LAN's. In *ACM SIGCOMM 1994*, pages 212–225, London, UK, August 31–September 2 1994.

[31] Dragos Niculescu and Badri Nath. Ad-hoc positioning system(aps). In *Proceedings of IEEE GLOBECOM*, November 2001.

[32] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5), October 1997.

[33] Jeffrey Hightower and Gaetano Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[34] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, October 2000.

[35] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for

very small devices, April 2000. Technical Report 00-729, University of Southern California.

[36] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In *IEEE INFOCOM*, volume 3, pages 1655–1663, 2001.

[37] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks1. In *Proc. of MobiCom 203*, Sept. 2003.

[38] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the Tenth ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Philadelphia, PA, September 2002.

[39] Lingxuan Hu and David Evans. Localization for mobile sensor networks. In *Proceedings of MobiCom 2004*, September 2004.

[40] Charles E. Perkins, Elizabeth M. Royer, and Samir R. Das. Ad hoc on-demand distance vector routing. In *October 99 IETF Draft*.

[41] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Computer Communication Review*, pages 234–244, Oct 1994.

[42] Niwat Thepvilojanapongy, Yoshito Tobez, and Kaoru Sezaki. SHR: Stateless hierarchical routing for dynamic sensor networks. In *The first International Workshop on Networked Sensing Systems (INSS '04)*, 2004.

[43] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, May 2001.

[44] J. Newsome and D. Song. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *The First ACM Conference on Embedded Networked Sensor Systems (Sensys03)*, November 2003.

[45] G. TOUSSAINT. *Pattern Recognition*, chapter The relative neighborhood graph of a finite planar set, pages 261–268. 1980.

[46] Karim Seada, Ahmed Helmy, and Ramesh Govindan. On the effect of localization errors on geographic face routing in sensor networks. In *IPSN '04*, April 2004.

[47] David M. Nicol, Michael E. Goldsby, and Michael M. Johnson. Simulation analysis of virtual geographic routing. In *Proceedings of the 2004 Winter Simulation Conference*, December 2004.

[48] Huan Li, Prashant Shenoy, and Krithi Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. *UMASS CMPSCI Technical Report TR04-91*, 2004.

[49] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-realtime environment. *Journal of the ACM*, 20(1), 1973.

[50] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: Operating system support for multimedia applications. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, May 1994.

[51] David Stewart and Michael Barr. Introduction to rate monotonic scheduling. http://www.netrino.com/Publications/Glossary/RMA.html.

[52] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. *Proceedings of the seventh annual international conference on Mobile computing and networking(Mobicom 2001)*, 2001.

[53] Huan Li, Prashant Shenoy, and Krithi Ramamritham. Scheduling communication in real-time sensor applications. *Proceedings of the Tenth IEEE Real/Time Embedded Technology and Applications Symposium (RTAS'04)*, May 2004.

[54] Sameer Tilak, Nael Abu-Ghazaleh, and Wendi Heinzleman. Infrastructure tradeoffs in sensor networks. *ACM Workshop on Sensor Networks and Applications (WSNA'02), Held in Conjunction with MobiCom 2002*.

[55] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. CODA: Congestion detection and avoidance in sensor networks. pages 266–279, Los Angeles, CA, November 2003.

[56] JaeWon Kang, Yanyong Zhang, and Badri Nath. Adaptive resource control scheme to alleviate congestion in sensor networks. In *the First Workshop on Broadband Advanced Sensor Networks*, 2004.

[57] 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*, 2002.

[58] I. Aad and C. Castelluccia. Differentiation mechanisms for ieee 802.11. In *IEEE INFOCOM 2001*, pages 1044–1056, April 2001.

[59] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/, 2004.

[60] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.