22

Real-Time Communication for Embedded Wireless Networks

Marco Caccamo
University of Illinois at Urbana-Champaign

Tarek Abdelzaher
University of Illinois at Urbana-Champaign

22.1 Introduction

In recent decades, computer systems have been embedded into physical environments for automated real-time monitoring and control. This trend will continue, and even expand, to improve and secure our quality of life in many areas such as defense, emergency rescue, biosensing, pervasive computing, and assisted living. Analyzing application scenarios such as (1) cooperative mobile robots for rescue activities and (2) wireless sensor networks for surveillance/biosensing, some important observations can be made. First, the system requires *end-to-end real-time performance* to ensure emergency response within bounded delay; second, mobility and ease of deployment are mandatory requirements in the described scenarios; hence, wired backbone or wireless links with infrastructure are not viable options. Finally, the described applications are mainly event driven and the runtime workload is dynamic, hence, real-time data flows need to be established on demand, they require bounded delay, low jitter, and guaranteed data throughput. Since timeliness is a property tightly related to the way each shared resource (such as the wireless medium) is managed at a low level, this chapter will first cover protocols and challenges related to the medium access control (MAC) layer, followed by an overview of a light-weight real-time communication architecture (RAP), a delay-sensitive network-layer protocol (SPEED), and a notion of network capacity to quantify the ability of the network to transmit information in time.

MAC protocols have been widely studied in wireless networks; however, most MAC schemes are primarily designed to achieve a high-average throughput and fairness among multiple users; real-time guarantee and jitter control are usually considered secondary goals and several protocol extensions exist that is able to provide some degree of quality of service (QoS). Focusing the attention on MAC protocols that provide real-time or QoS support, it is possible to broadly categorize them into two classes [14]: *synchronous schemes* that require global time synchronization among different nodes and *asynchronous schemes*, which do not require such support.

Synchronous schemes include Implicit-EDF [8], TBMAC [10], Cluster TDMA [17], Cluster Token [16], and MAX [22]. All these protocols work by having nodes that implicitly agree on a transmission slot assignment. The agreement requires both time synchronization and either a regular network structure or some form of clustering. IEEE 802.11 Distributed Coordination Function (DCF) is a widely used standard asynchronous protocol, but its ability to provide reserved bandwidth to differentiated multihop traffic is poor [15]. Several schemes have been developed to improve over DCF by either providing support for differentiated traffic categories (see real-time MAC [6] and DCF with priority classes [11]), or by reducing collisions in multihop networks (see MACA/PR [18]). Though these schemes are easy to implement over the IEEE 802.11 protocol, owing to the nature of 802.11 random backoff they cannot provide delay bounds for real-time traffic.

Among the class of asynchronous schemes, there are several protocols based on Black-Burst (BB) [26,27] and they have been proposed to support real-time traffic [13,24]. However, all such protocols assume a fully connected network, and are not easily extensible to a multihop environment. Robust and implicit earliest deadline first (RI-EDF) [9] is able to provide real-time guarantee, low jitter, and high bandwidth in single-hop networks by relying on a shared packet schedule among all nodes. As such, RI-EDF is also hard to extend to large networks. The implicit contention scheme adopted by RI-EDF and Implicit-EDF introduces small overhead outperforming BB in terms of achievable network throughput [8]. Out-of-band signaling has been used for channel reservations in multihop networks in different works [23,28,32]. Although these protocols use multiple wireless channels to provide better temporal QoS, they need a transceiver that is able to simultaneously listen to two different channels at once, which is not practical in resource-constrained embedded systems.

22.2 Basic Concepts for Predictable Wireless Communication

When deploying a real-time wireless network (single or multihop), both the physical layer and MAC can greatly affect its temporal predictability. In fact, at physical layer radio-frequency (RF) interference, large-scale path loss and fading cause adverse channel conditions by reducing *signal-to-noise ratio* (SNR) of the wireless communication. When the SNR is lower than a certain threshold, the *bit error rate* (BER) of the wireless communication rises over the acceptable limit, thus disrupting the wireless connection. The key to maintain wireless communication under adverse channel conditions is to provide as high SNR as possible. To achieve this, a promising solution lies in the state-of-the-art *direct sequence spread spectrum* (DSSS) [30,31] technology, which allows trade-offs between data throughput versus SNR. It is easy to understand that high BER would cause packet retransmissions and most probably deadline misses. While the physical layer is out of the scope of this chapter, the following sections will focus on MAC techniques, will introduce a light-weight real-time communication architecture, and a notion of network capacity for real-time wireless networks. To make the idea of real-time wireless networks work, two main problems need to be addressed at the MAC layer:

- Packet collisions on wireless channel and
- Priority inversions when accessing the wireless medium

A packet collision occurs when two nearby nodes decide to transmit a packet at the same time resulting in a collision; priority inversion occurs each time a node carrying a high-priority packet loses wireless medium contention against another node that has a lower-priority packet: an ideal real-time wireless network should not experience any packet collision and should experience only bounded priority inversions.

To avoid packet collisions and mitigate the priority inversion problem caused by the multihop nature of large wireless networks, a prioritized medium access scheme should be used, for instance, a suitable class of protocols is the one based on BB [26]. The original BB approach was designed to achieve fair channel sharing in wireless LAN; to achieve real-time performance, the BB channel contention scheme can be used to differentiate among different real-time priority levels. In a single-hop environment, packet collisions can be avoided by using standards such as IEEE 802.15.4 and a network coordinator; similarly, IEEE 802.11 PCF implements the notion of contention-free period to avoid conflicts and enforce a resource reservation mechanism. Both the standards allow to provide temporal QoS. In the next section, RI-EDF will be presented as real-time single-hop MAC scheme; it is able to provide real-time guarantee, low jitter, and high bandwidth by relying on a shared packet schedule among all nodes. Compared to existing standards, RI-EDF has the advantage of having lower overhead and increased robustness; however, this access scheme is suitable for mainly periodic and semiperiodic real-time traffic.

In a multihop environment, temporal predictability is a challenging task and an open research problem the real-time and network communities are looking at. In the subsequent sections, we shall briefly describe routing protocols that address time and distance constraints. We shall also describe appropriate higher-level communication abstractions. Finally, a notion of network capacity will be derived to quantify the ability of the network to transmit information in time. As a final remark, it is important to notice that every wireless communication setting is always subject to the unpredictability of the wireless medium despite any efforts of building an ideal collision-free and priority inversion-free wireless MAC; as such, all the protocols described in this chapter cannot guarantee hard real-time constraints. Instead, they focus on providing soft real-time guarantees. Therefore, all performance bounds and real-time guarantees expressed in the following sections are provided subject to the assumption that the wireless medium is not affected by jamming or electromagnetic interference (EMI).

22.3 Robust and Implicit Earliest Deadline First

The RI-EDF protocol [9] derives a schedule for the network by using EDF [19]. In fact, nodes transmit according to the obtained schedule, implicitly avoiding contention on the medium. A node transmits when it receives the entire packet train from the previous node or receives a recovery packet and is its turn in the schedule for sending a packet; otherwise, a node transmits when it must recover the network.

22.3.1 Terminology and Assumptions

A network consists of q nodes labeled N_i, where $i = 1, 2, \ldots, q$. The maximum packet size, in terms of transmission time (seconds), is denoted by θ. M_j is a periodic message characterized by (m_j, T_j, i), where m_j is the message length, T_j the period of the message, and i the index of the transmitting node. Both m_j and T_j are measured in seconds. Aperiodic messages are denoted by the same tuple, where T_j represents the minimum interarrival time between messages. Aperiodic messages can be scheduled using an aperiodic server, such as a polling server [20]. In this way, each server can be treated as a periodic message for the purpose of schedulability analysis. $\mathcal{M}$ is the set of all periodic messages and aperiodic servers in the system.

The network schedule is derived from $\mathcal{M}$ for a single hyperperiod using the rules of EDF. The hyperperiod, H, is the least common multiple of all the message periods. Nodes are prioritized according to i. A lower i denotes a higher priority. During schedule derivation, ties are broken in favor of the node with the highest priority. The network schedule is represented by a sequence of *packet trains*. Each packet train is a sequence of contiguous packets transmitted by a single node and characterized by the index of sender node and duration of the packet train itself. A packet train includes one or more packets, each denoted by a packet number p_j. Note that the length of the last packet of each packet train is less than the maximum size θ, if the duration of the packet train is not a multiple of the maximum packet size.

More formally, the kth packet train of the network schedule within a single hyperperiod is denoted by $PT_k(t_s, t_f, i)$, for $k = 0, \ldots \mathcal{K}$, where t_s denotes the start time of the packet train, t_f the finish time of the packet train, and i the index of the transmitting node. The packet train currently being transmitted by the

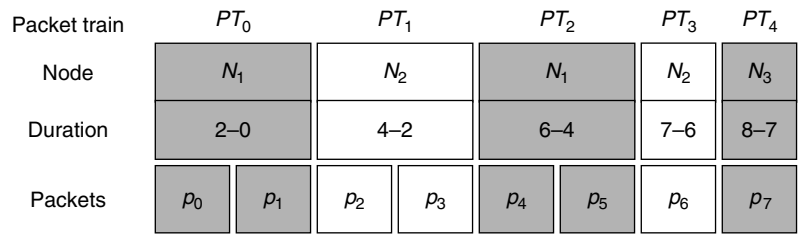

FIGURE 22.1 Example of schedule.

current node, N_{curr}, is denoted by PT_{curr}. PT_{prev} denotes the previously transmitted packet train by node N_{prev}. The remaining time a node has to send PT_{curr} is its *budget*. The budget for N_i is defined as a function of time t and is denoted by $C_i(t)$. At the beginning of a packet train transmission, $C_i(t) = t_f - t_s$. At any time t during transmission, $C_i(t) = t_f - t$ is the remaining time for a packet train transmission. Each packet has an identifying *packet number* p_j, where j is incremented starting at 0 within the hyperperiod. Each node has an *internal state* s_i given by the packet number p_j of the last correctly received or transmitted packet. Every node maintains its own internal state. Nodes utilize the network schedule to determine, which node is next to transmit. Figure 22.1 summarizes the network schedule definition and provides the schedule for the example in Section 22.3.2.

RI-EDF is robust in the event of packet losses and node failures; in fact, assuming a fully linked network, it always guarantees a conflict-free schedule and has a distributed recovery mechanism that allows to recover the network communication in the event of node failures. This protocol does not consider security aspects such as nodes transmitting malicious packets, or other forms of Byzantine failures. Packet data are always trusted. The timer set by each node to perform network recovery is the *recovery timer*. The length of the recovery timer for node N_i is proportional to the static priority of the node.

The following assumptions are made about the network: (1) The network is fully linked; that is, every node is within direct transmission range of every other node; (2) All nodes have knowledge of current network schedule or the knowledge of $\mathcal{M}$ (the set of all periodic messages and aperiodic servers in the system); and (3) Nodes that do not successfully receive a transmitted packet are still capable of using carrier sense to detect a busy channel.

22.3.2 The RI-EDF Protocol

This section describes the RI-EDF protocol. Throughout this section, the schedule of Figure 22.1 is used as an example. The example schedule consists of three nodes $\{N_1, N_2, N_3\}$, sending the messages $M_0 = (2, 4, 1)$, $M_1 = (3, 8, 2)$, and $M_2 = (1, 8, 3)$ on the network. The schedule is derived using the rules of EDF. Using node priority to break ties, the schedule, expressed in terms of packet trains is $\{PT_0 = (0, 2, 1)$, $PT_1 = (2, 4, 2)$, $PT_2 = (4, 6, 1)$, $PT_3 = (6, 7, 2)$, $PT_4 = (7, 8, 3)\}$. For simplicity, the examples assume a constant packet size with $\theta = 1$. RI-EDF uses two mechanisms to ensure robustness in cases of varied packet size, node failure, and packet loss: *budget* and *recovery*.

Budget. A node that completes its transmission early causes schedule drift. The subsequent node may not have a packet prepared for transmission and thus forced to forgo transmission. To prevent schedule drift, RI-EDF uses a *budget*. The unused budget of one node may be forwarded to the immediately following node in the schedule. A node may have unused budget if its periodic data transmission is less than the scheduled amount. The forwarded budget may then be used by the subsequent node to transmit any aperiodic messages until its periodic messages are ready. Δ denotes the budget forwarded to one node by its previous node. Unused budget cannot be saved by a node for use in transmitting its later packet trains.

To support the budget mechanism, the RI-EDF protocol uses five fields in each *data packet*: (1) Type indicates that this is a data packet; (2) Packet train completion the final packet of a packet train; (3) Forwarded budget the forwarded budget Δ conveyed by the previously transmitting node; (4) Packet

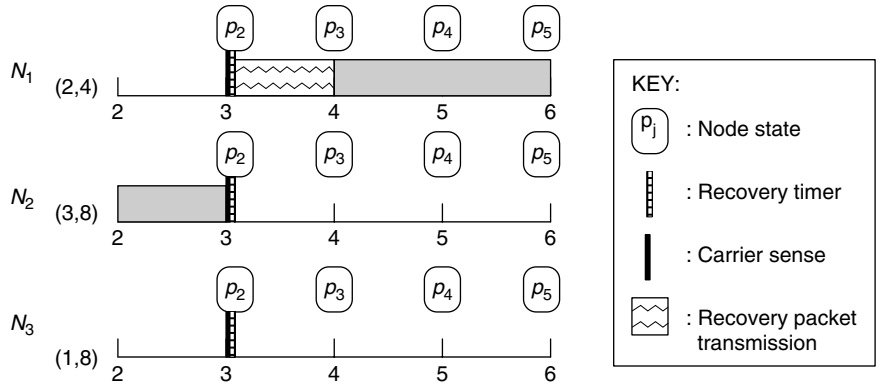

FIGURE 22.2 RI-EDF utilizes recovery mechanism to avoid network stall in the event of N_2 failure.

number the packet number p_j within the hyperperiod; and (5) Payload the variable-length data. When PT_k begins transmission, its budget $C_k(t)$ is initialized to $(t_f - t_s)$.

Recovery. Care must be taken to guarantee that packet transmissions continue, even in the presence of node failures or packet losses. For this purpose, a recovery mechanism is used. Upon the transmission or reception of a packet, each node performs carrier sense on the medium. If the medium is busy, no recovery is performed. The maximum idle time permitted is t_{idle}. If the medium is idle for t_{idle}, it is assumed that transmissions have stalled and recovery must be performed. Each node sets a recovery timer for a time proportional to its static priority. If the recovery timer expires on a given node, the node sends a recovery packet using its own internal state. Nodes that successfully receive the recovery packet update their internal state with the recovery packet number and can thus determine N_{curr}.

To support the recovery mechanism, RI-EDF uses three fields in each *recovery packet*: (1) Type indicates that this is a recovery packet; (2) Packet number the packet number p_j within the hyperperiod; and (3) Payload the variable-length data. Depending on implementation decisions of the designer, the payload can be zero-length or it can be used to send aperiodic data. Notice the recovery operation in Figure 22.2 in which N_2 fails to transmit one of its packets. N_2 transmits successfully at time $t = 2$. Nodes N_1 and N_3 successfully receive packet p_2 and update their internal state. At time $t = 3$, N_2 fails to transmit. Each node performs carrier sense and sets a recovery timer after t_{idle}. Node N_1 has the highest-recovery priority, and its recovery timer expires first. It then sends a recovery packet using its internal state to indicate that p_3 should have been sent. After recovery, N_1 transmits packets p_4 and p_5 as normal.

22.3.2.1 RI-EDF Rules

Figure 22.3 summarizes the rules of the RI-EDF protocol as a finite state machine. The five states are (1) Carrier Sense I; (2) Carrier Sense II; (3) Receive; (4) Transmit; and (5) Recovery.

1. *Carrier Sense I:* the initial state. This state is also reached when a node has completed transmission and it is not N_{curr}, or because a busy medium has been detected.
   Disable recovery timer. Perform carrier sense.
2. *Carrier Sense II:* this state is reached upon the detection of an idle medium for t_{idle}.
   Set recovery timer for a time proportional to the static priority of N_i. If a busy medium is detected, return to State 1, Carrier Sense I. If recovery timer expires, proceed to State 5, Recovery.
3. *Receive:* this state has been reached because a packet has been received by the node.
   Record a MAC-layer timestamp $\bar{t}$. Set the node's internal state to the received packet number. If the completion bit is set in the received packet, and this node is N_{curr}, set the budget to any forwarded budget plus the transmission time of PT_{curr}, that is, $C_{curr}(t) = \Delta + t_f - t_s$. Proceed to State 4, Transmit. Otherwise, return to State 1, Carrier Sense I.

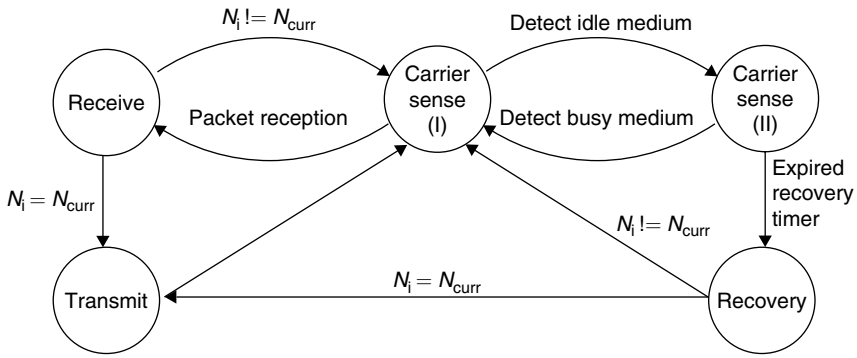

FIGURE 22.3 RI-EDF protocol.

4. *Transmit:* this state has been reached because this node, N_i, is N_{curr}, the currently transmitting node. *If there is a forwarded budget* Δ: Transmit aperiodic messages whose total transmission time is no greater than Δ. If there is a forwarded budget and no aperiodic messages remain, transmit one or more zero-payload data packets to hold the medium for Δ. While holding the medium or transmitting aperiodics, the packet number transmitted is the internal state of the node and the internal state is not altered. *If there is no forwarded budget, or the forwarded budget is exhausted:* The budget is $C_{curr}(t)$. The packet number is equal to the internal state of this node plus one. Send periodic packets, ensuring that the packet transmission time does not exceed the allocated budget of the node. After each packet transmission, set the internal state to the most recently transmitted packet number. The final packet in the packet train sets the completion bit and conveys any forwarded budget to the next node. *If there are no periodics:* Transmit a zero-payload data packet to forward this node's budget. At transmission completion, return to State 1, Carrier Sense I.
5. *Recovery:* this state has been reached because a recovery packet must be transmitted. All forwarded budget is discarded. The packet number is equal to the internal state of this node plus one. Transmit a *recovery* packet. Update the internal state of the recovery node to this packet number. If after updating, the recovery node is now N_{curr}, proceed to State 4, Transmit. Otherwise, return to State 1, Carrier Sense I.

To use the proposed wireless MAC scheme, the network has to be initialized by building the network schedule, disseminating it among all the nodes, and finally by starting the transmission of the first packet. A simple way of initializing the network is to download offline the initial network schedule on all nodes; in fact, as soon as the first node is turned on, it starts to send recovery packets and its reserved packets until one by one all the other nodes are turned on. As more nodes are activated, less recovery packets are transmitted on behalf of the missing nodes. Eventually, all the nodes are turned on and the network schedule is followed by all of them. Basically, the recovery mechanism takes care of bootstrapping the network independently of the activation order of the nodes.

22.3.2.2 Protocol Properties

Property 22.1
Assuming a fully linked network, RI-EDF maintains a conflict-free schedule in the presence of node failures and packet losses without a centralized scheduler.

Proof
In the absence of packet losses and node failures, every node is implicitly aware of the schedule. At every scheduling instant (the instant of packet reception), there is a single node trying to send a packet in the channel. Thus, a conflict-free schedule is maintained.

Packet losses and node failures: Consider a packet loss at a node N_k. That is, N_k does not receive the previously transmitted packet, p_j. Let N_p be the live node, which has the highest static priority. Let N_q be the node, which is scheduled for transmission after the dropped packet.

Case 1: N_q is alive and it has received p_j. N_q becomes N_{curr}, and thus has the highest priority to transmit. Every other node, including N_k, will refrain from transmission after it detects a busy medium. The schedule is unaltered and still conflict-free.

Case 2: $N_q = N_k$. Thus, the node to transmit next did not receive p_j successfully. As a result, N_p becomes the node with the highest static priority to recover the schedule. N_p transmits the sequence number in the recovery packet. When N_p recovers the schedule, every other node, including N_k, will detect a busy medium and refrain from transmission. Thus, the schedule is maintained conflict-free.

An analogous argument holds in the case of node failure at N_k. Note that a conflict-free schedule is maintained since detection of the idle channel is the synchronization point for all nodes belonging to a fully linked network.

Property 22.2

RI-EDF has no central point of failure.

This property follows immediately from the fact that all nodes are involved in the recovery mechanism.

Schedulability Analysis: In the absence of packet loss and node failures, the schedulability of a set $\mathcal{M}$ of real-time periodic messages and aperiodic servers (denoted by (m_j, T_j, i), where $j = 1, 2, \ldots, n$) can be tested by the following sufficient condition [5]:

$$\forall j, \quad 1 \le j \le n \quad \sum_{k=1}^{j} \frac{m_k}{T_k} + \frac{\theta}{T_j} \le 1$$

which assumes that all the messages are sorted by increasing periods, that is, $T_i \le T_j$ only if $i < j$. It is worth noting that the blocking time of each message is equal to the maximum packet size θ since packets are nonpreemptable.* Under the assumption that the above schedulability condition is verified, that there are no packet losses and the network is fully linked (ideal case), the delay of all real-time periodic messages is bounded and it is easy to verify that the maximum delay of each message M_j never exceeds twice its period $2T_j$. In fact, according to EDF, two consecutive instances of the same periodic message can never be scheduled more than two periods apart otherwise the schedulability guarantee would fail.

22.3.3 Power-Aware RI-EDF

In some applications targeted by the RI-EDF protocol, low power consumption is mandated. In the RI-EDF protocol, the channel is in constant use, even in cases where there are no packets ready for transmission. As a result, a power-aware extension is introduced for the RI-EDF protocol. This extension is based on differentiating between nodes that are *sources* or *sinks*. Sources are those nodes that transmit data. Sinks are those nodes that are interested in the actual *data* sent in each packet; data that will be processed later in some fashion. Sinks may also transmit data. To support this extension, two additions are made. First, a field is added to the packet headers to denote whether a packet was transmitted by a sink or source node. Second, sinks must have a periodic beacon packet in the schedule that may be optionally transmitted each hyperperiod to solicit data from the sources. Using these additions, the power-aware RI-EDF extension rules are as follows:

After a source node performs recovery, its recovery mechanism is disabled and it is no longer able to recover the network. Once its recovery mechanism is disabled, a source node can send up to $\mathcal{P}$ data packets before it may no longer transmit. $\mathcal{P}$ is based on the design decisions of the network developer. Recovery is

*This real-time guarantee does not hold in the presence of packet loss since RI-EDF does not include reservations for packet retransmissions. Ref. 9 summarizes the effects of packet loss and hidden nodes on RI-EDF, and experimentally shows a gradual degradation of performance.

reenabled in sources only when they receive a packet sent by a sink. Sinks have a higher static priority than the sources in the event that recovery must take place, allowing sources to reenable their recovery mechanisms as quickly as possible. In addition, sinks do not disable their recovery mechanisms unless they want to sleep. This extension has two advantages. First, if a sink does not require any data, sources will stop transmitting, thus avoiding power consumption for unnecessary transmissions. Second, source nodes that may move out of the range of sinks will stop transmitting after a maximum of $\mathcal{P}$ packets after which the sources may sleep. In addition, sources may sleep at any time and wake up periodically to determine if they can hear a sink in which case they will immediately rejoin the schedule. This sleeping conserves power in the sources.

The current consumption of TinyOS and power-aware RI-EDF protocols were compared. Five sources generated real-time random traffic requests for throughput ranging from 200 to 1000 bytes/s. Each source disabled its radio between requests. The state of the radio was monitored, observing the time it was in a transmit or receive state. Using the time spent in each of these states, current consumption was estimated for each protocol. As summarized in Tables 22.1 and 22.2, at low channel utilization, RI-EDF consumed more current than the TinyOS protocol. With increased utilization, RI-EDF consumes less current while transmitting more packets than TinyOS. This is a direct result of RI-EDF's fewer dropped packets due to reduced contention on the medium. This experiment also demonstrates that RI-EDF experiences very few deadline misses compared to TinyOS. In particular, RI-EDF experienced deadline misses only when the wireless channel was overloaded. Finally, in Ref. 9 experiments were performed to compare the communication throughput of RI-EDF versus the TinyOS MAC protocol. In the throughput experiments, seven Motes were employed, all within the range of each other. An eighth Mote was used to monitor all packet transmissions. The packet size was chosen such that no budget would be forwarded during normal operation. The RI-EDF protocol achieves transmission rates close to the maximum speed of the radio (19.2 Kbps). Although RI-EDF has more overhead, data throughput is greater for all tested payload sizes. For the largest payload size, RI-EDF has an additional 10.8% overhead over TinyOS, including the overhead required for the Dynamic Schedule Update extension described in Section 22.3.4. Despite the overhead, RI-EDF allows a 26% increase in transmitted data.*

TABLE 22.1 Tiny OS Current Consumption

Target Bps	Actual Bps	% Collisions	% Missed Deadlines	% Time in		Current (mA)
				Trans.	Recv.	
200	199.0	0.33	0	3.66	7.52	1.338
400	392.9	0.81	0	7.09	19.22	3.037
600	597.0	1.80	0	10.67	36.89	5.334
800	780.9	3.62	29.28	13.60	70.12	9.017
1000	820.9	4.32	69.60	14.22	81.79	10.240

TABLE 22.2 RI-EDF Current Consumption

Target Bps	Actual Bps	% Collisions	% Missed Deadlines	% Time in		Current (mA)
				Trans.	Recv.	
200	204.9	0	0	3.77	11.60	1.746
400	404.8	0	0	7.42	24.46	3.595
600	602.1	0	0	11.03	39.82	5.675
800	802.1	0	0	14.68	55.65	7.809
1000	1001.3	0	0.25	18.31	72.25	10.012

*The restricted packet sizes of MICA2 Motes skews the throughput statistics against RI-EDF owing to its 2-byte packet header overhead. This overhead would be insignificant on implementations permitting larger packet sizes.

22.3.4 Dynamic Schedule Updates

Dynamically updating the RI-EDF schedule is necessary in applications where a node's participation in the network is changing. Owing to the mobility of some networks, certain applications may be interested in removing nodes from or adding nodes to the schedule. Other applications may be interested in updating the schedule based on changing demands of existing nodes. A simple mechanism to update the system schedule is presented here. The mechanism for adding new nodes utilizes a reservation for a packet train at the end of a hyperperiod, or once in several hyperperiods. This packet train reservation, denoted as PT_{res}, is dedicated to node addition requests and replies, schedule update requests, and new schedule announcements.

Requests are serviced by a leader node, the node with the highest static priority in the system.* This node could also be elected by approaches in Refs. 29 and 33, or even simply using node address-based election. Nodes intending to join contend to transmit during PT_{res}. Nodes transmit a random request identifier, which will be used in the acknowledgment to differentiate between requests. All requests are processed in FIFO order. The leader node replies to the join requests with a positive or negative acknowledgment, depending on if the schedule can accommodate the request. The priorities for transmission during the PT_{res} slot are as follows, with 1 being the highest static priority: (1) *Schedule announcements after (a) join requests and (b) dynamic schedule update requests*. This priority is mapped to the priority of the leader node. (2) *Dynamic schedule update requests* and *join requests.* This new static priority level is called *join priority*. (3) *Unrequested schedule announcements*. This priority is mapped to the standard fixed priorities of the nodes. If there are no schedule updates or join requests, the node with the highest fixed priority that is alive, including the leader, will transmit this *unrequested schedule announcement* to announce the current schedule.

This mechanism can also be used for removing nodes from the schedule. Nodes make an exit request during the reserved slot in the hyperperiod. The leader node acknowledges the request, and the schedule is recomputed and announced appropriately. The schedule can also be updated without receiving requests during PT_{res}. Existing nodes can make requests to change their utilization in the schedule.

This mechanism employs a *new schedule* bit, or *NS* bit in the data packet header to indicate that a newly derived schedule will be used starting at the next hyperperiod. Once the leader node acknowledges a request and announces a new schedule, all nodes set the *NS* bit in their subsequent data packet transmission until the new schedule begins in the following hyperperiod. This is done to increase the chance that all nodes receive a packet with a set *NS* bit before the start of the next hyperperiod at which point the new schedule begins. Also added to the data packet header is the *schedule identifier*. The schedule identifier is a value identifying the schedule with which the transmitted data packet is associated. Each time a new schedule is announced by the leader node, an accompanying schedule identifier is included. Each node keeps an internal copy of the current schedule identifier. This internal copy of the schedule identifier is updated only when a node successfully receives the accompanying schedule announcement during PT_{res}. The *NS* bit and schedule identifier help to mitigate the problems that arise from a protocol such as RI-EDF, which does not employ consensus. These two header fields help to insure that nodes do not transmit if they do not have the correct schedule.

22.4 Higher-Level Real-Time Protocols for Sensor Networks

This section briefly reviews higher-level communication protocols in sensor networks. An earlier draft was presented at the DDRTS 2003 workshop [3].

22.4.1 RAP: Real-Time Distance-Aware Scheduling

Message communication in sensor networks must occur in bounded time, for example, to prevent delivery of stale data on the status of detected events or intruders. In general, a sensor network may simultaneously carry multiple messages of different urgency, communicated among destinations that

*The leader node is equivalent to a sink node described in Section 22.3.3.

are different distances apart. The network has the responsibility of ordering these messages on the communication medium in a way that respects both time and distance constraints.

A protocol that achieves this goal is called RAP [21], developed at the University of Virginia. It supports a notion of packet velocity and implements velocity-monotonic scheduling (VMS) as the default packet scheduling policy on the wireless medium. Observe that for a desired end-to-end latency bound to be met, an in-transit packet must approach its destination at an average velocity given by the ratio of the total distance to be traversed to the requested end-to-end latency bound. RAP prioritizes messages by their required velocity such that higher velocities imply higher priorities. Two flavors of this algorithm are implemented. The first, called *static velocity-monotonic scheduling*, computes packet priority at the source and keeps it fixed thereafter regardless of the actual rate of progress of the packet toward the destination. The second, called *dynamic velocity-monotonic scheduling*, adjusts packet priority *en route* based on the remaining time and the remaining distance to destination. Hence, a packet's priority will increase if it suffers higher delays on its path and decrease if it is ahead of schedule.

To achieve consistent prioritization in the wireless network, not only does one need priority queues at nodes, but also a MAC layer that resolves contention on the wireless medium in a manner consistent with message priorities. The authors of RAP [21] used a scheme similar to Ref. 1 to prioritize access to the wireless medium. The scheme is based on modifying two 802.11 parameters, namely the DIFS counter and the backoff window, such that they are priority-aware. An approximate prioritization effect is achieved by letting backoff depend on the priority of the outgoing packet at the head of the transmission queue. Alternatively, some version of BB could have been used.

A detailed performance evaluation of this scheme can be found in Ref. 21. It is shown that velocity-monotonic scheduling substantially increases the fraction of packets that meet their deadlines taking into consideration distance constraints.

22.4.2 SPEED: Enforcement of Velocity Constraints

RAP leads the idea that a network can support multiple predefined velocities. An application chooses a velocity level for each message. The network guarantees that the chosen message velocity is observed with a very high probability as long as the message is accepted from the application. A network-layer protocol with the above property, called SPEED [12], has been developed at the University of Virginia. The protocol defines the velocity of an in-transit message as the rate of decrease of its straight-line distance to its final destination. Hence, for example, if the message is forwarded away from the destination, its velocity at that hop is negative.

The main idea of SPEED is as follows. Each node i in the sensor network maintains a neighborhood table that enumerates the set of its one-hop neighbors. For each neighbor, j, and each priority level, P, the node keeps a history of the average recently recorded local packet delay, $D_{ij}(P)$. Delay $D_{ij}(P)$ is defined as the average time that a packet of priority P spends on the local hop i before it is successfully forwarded to the next-hop neighbor j. Given a packet with some velocity constraint, V, node i determines the subset of all its neighbors that are closer to the packet's destination. If L_{ij} is the distance by which neighbor j is closer to the destination than i, the velocity constraint of the packet is satisfied at node i if there exists some priority level P and neighbor j, such that $L_{ij}/D_{ij}(P) \geq V$. The packet is forwarded to one such neighbor nondeterministically. If the condition is satisfied at multiple priority levels, the lowest priority level is chosen. If no neighbor satisfies the velocity constraint, we say that a local deadline miss occurs.

A table at node i keeps track of the number of local deadline misses observed for each velocity level V. This table is exchanged between neighboring nodes. Nodes use this information in their forwarding decisions to favor more appropriate downstream hops among all options that satisfy the velocity constraint of a given packet. No messages are forwarded in the direction of nodes with a high miss-ratio. The mechanism exerts backpressure on nodes upstream from congested areas. Congestion increases the local miss-ratio in its vicinity, preventing messages from being forwarded in that direction. Messages that cannot be forwarded are dropped thus increasing the local miss-ratio upstream. The effect percolates toward the source until a node is found with an alternative (noncongested) path toward the destination, or the source

is reached and informed to slow down. The mentioned scheme is therefore effective in exerting congestion control and performing packet rerouting that guarantees the satisfaction of all velocity constraints in the network at steady state [12]. The protocol is of great value to real-time applications where different latency bounds must be associated with messages of different priority.

22.4.3 Entity-Aware Transport

Although RAP and SPEED allow velocity constraints to be met, the abstractions provided by them are too low level for application programmers. A transport layer is needed whose main responsibility is to elevate the degree of abstraction to a level suitable for the application. One such transport layer is described in Ref. 7. The authors propose a transport layer in which connection endpoints are directly associated with events in the physical environment. Events represent continuous external activities, such as the passage of a vehicle or the progress of a fire, which is precisely what an application might be interested in. By virtue of this layer, the programmer can describe events of interest and logically assign "virtual hosts" to them. Such hosts export communication ports and execute programs at the locations of the corresponding events. The programmer is isolated from the details of how these hosts and ports are implemented. When an external event (e.g., a vehicle) moves, the corresponding virtual host migrates with it transparently to the programmer.

We call the virtual host associated with an external event of interest an *entity*. Sensor nodes that can sense the event are called *entity members*. Members elect an *entity leader* that uniquely represents the entity and manages its state. Hence, an entity appears indivisible to the rest of the network. The fact that it is composed of multiple nodes with a changing membership is abstracted away. It is key to maintain a unique entity (with a single leader) to describe any specific external object or activity. This is called *entity uniqueness*. It is needed to maintain the abstraction that communication endpoints are logically attached to mobile targets.

An evaluation of this architecture reveals that entity uniqueness is maintained as long as the target event moves in the environment at a speed slower than half the nodes' communication radius per second [7]. For example, if sensor nodes can communicate within a 200-m radius, the transport layer can correctly maintain endpoints attached to targets that move as fast as 100 m/s (i.e., 360 km/h). The combination of this transport layer and the guaranteed velocity protocols described earlier provides invaluable support to real-time applications. For example, communication regarding moving targets can be made to proceed in the network at a velocity that depends on target velocity itself. Hence, positions of faster targets, for example, can be reported quicker than those of slower ones.

22.5 Real-Time Capacity of Wireless Networks

The protocols described above attempt to provide real-time communication in sensor networks such that data are delivered on time. An interesting theoretical question is to quantify the capacity of a sensor network to deliver information by deadlines. This is called *real-time capacity*.

In this section, the theoretical foundations and approach are presented for deriving expressions of real-time capacity of sensor networks. We begin by defining real-time capacity more formally. Real-time capacity, first introduced in Ref. 2, refers to the total byte-meters that can be delivered *by their deadlines*. To make the capacity expressions independent of the details of the workload (such as the deadline values themselves), a normalized bound is defined that quantifies the total byte-meters that can be delivered for each time unit of the relative deadline. To illustrate the notion of real-time capacity, consider a network with two flows, A, and B. Flow A must transfer 1000 bytes a distance of 50 m (i.e., a total of 50,000 byte-meters) within 200 s. It is said to have a real-time capacity requirement of $50{,}000/200 = 250$ byte-meters/s. Flow B must transfer 300 bytes a distance of 700 m within 100 s. Its capacity requirement is thus $300 * 700/100 = 2100$ byte-meters/s. Hence, the total real-time capacity needed is $2100 + 250 = 2350$ byte-meters/s. An interesting question is whether one can establish the total real-time capacity a communication

network can support as a function of different network parameters, such that all flows meet their deadlines as long as their collective capacity requirements do not exceed the derived capacity bound.

22.5.1 The Single-Path Feasible Region

To compute the capacity bound, we first establish the schedulability condition for one path in the network. We then use it to find the total amount of network traffic that can meet deadlines. Consider a sensor network with multiple data sources and data sinks. Packets traverse the network concurrently, each following a multihop path from some source to some destination. Each packet T_i has an arrival time A_i defined as the time at which the sending application injects the packet into the outgoing communication queue of its source node. The packet must be delivered to its destination no later than time $A_i + D_i$, where D_i is called the relative deadline of T_i. Different packets may generally have different deadlines. We call packets that have arrived but whose delivery deadlines have not expired as *in-transit* packets. Each packet T_i has a transmission time C_i that is proportional to its length. This transmission time is incurred at each forwarding hop of its path.

Increasing the amount of data in the network or reducing some of the deadlines will decrease schedulability. A metric called *synthetic utilization* is defined that captures the impact (on schedulability) of both the resource requirements and urgency associated with packets. Each packet contributes an amount C_i/D_i to the synthetic utilization of each hop along its path in the interval from its arrival time A_i to its absolute deadline $A_i + D_i$. Observe that schedulability decreases with increased synthetic utilization. This leads to the idea that a bound on synthetic utilization may exist that separates schedulable from (potentially) unschedulable workloads.

At any time t, let $S(t)$ be the set of packets that are in-transit in the entire sensor network. Hence, $S(t) = \{T_i | A_i \leq t < A_i + D_i\}$. Let $S_j(t) \in S(t)$ be the subset of $S(t)$ that passes through node j. The synthetic utilization, $U_j(t)$ of node j is $\sum_{T_i \in S_j(t)} C_i/D_i$, which is the sum of individual contributions to synthetic utilization (on this node) accrued over all in-transit packets passing through that node.

Consider an arbitrary path P through the sensor network without loss of generality. Let us number the hops of that path $1, \ldots, N$ from source to destination. An interesting question is to find a function $g(U_1, \ldots, U_N)$ of the synthetic utilization of each hop along the path, such that the end-to-end deadlines of all packets transmitted along that path are met when $g(U_1, \ldots, U_N) \leq B$, where B is a constant bound.

The feasible region was computed in Ref. 4 for an arbitrary fixed-priority scheduling policy. To quantify the effect of the scheduling policy, a parameter α is defined. Intuitively, α represents the degree of *urgency inversion* in the priority assignment observed under the given scheduling policy. An urgency inversion occurs when a less-urgent packet (i.e., one with a longer relative deadline) is given an equal or higher priority compared to a more urgent one. Let us call them packet T_{hi} and T_{lo}, respectively. Formally, we define $\alpha = \min_{T_{\text{hi}} \geq T_{\text{lo}}} D_{\text{lo}}/D_{\text{hi}}$, which is the minimum relative deadline ratio across all priority-sorted packet pairs. In the absence of urgency inversion (e.g., for deadline monotonic scheduling), $\alpha = 1$. Otherwise, $\alpha < 1$. For example, if priorities are assigned randomly, $\alpha = D_{\text{least}}/D_{\text{most}}$, where D_{least} and D_{most} are the minimum and maximum relative deadlines in the packet set, respectively. Hence, the objective is to find a function $g(U_1, \ldots, U_N, \alpha)$, such that $g(U_1, \ldots, U_N, \alpha) \leq B$ implies that all packets are schedulable for the particular fixed-priority scheduling policy.

In was shown in Ref. 4 that the feasible region for such a scheduling policy is

$$\sum_{j=1}^{N} \frac{U_j(1 - U_j/2)}{1 - U_j} < \alpha \tag{22.1}$$

In particular, for deadline monotonic scheduling, $D_n/D_{\max} \geq 1$, or $\alpha = 1$. The feasible region of path P under deadline monotonic scheduling is therefore

$$\sum_{j=1}^{N} \frac{U_j(1 - U_j/2)}{1 - U_j} < 1 \tag{22.2}$$

22.5.2 Approximate Total Capacity

The results derived above represent exact sufficient conditions on path schedulability. In Ref. 2, these results were used to derive a network real-time capacity bound. A packet *always* meets its end-to-end deadline as long as Equation 22.1 holds for its path. The main contribution of Equation 22.1 lies in relating end-to-end delay to a bound on the sum of throughput-like metrics (synthetic utilizations). These metrics can now be related to real-time capacity, hence establishing the real-time capacity bound. In the following, we give a brief sketch of how the above results can be used to derive an approximate real-time capacity for the network.

From Equation 22.1, the average synthetic utilization U of a node on a communication path of length N satisfies

$$\frac{U(1-U/2)}{1-U} < \alpha/N \tag{22.3}$$

Solving for U, we get

$$U < 1 + \frac{\alpha}{N} - \sqrt{1 + \left(\frac{\alpha}{N}\right)^2} \tag{22.4}$$

Remember that, by definition, $U = \sum_i C_i/D_i$ over all in-transit packets through a node. Since multiplying the packet transmission time, C_i, by the channel transmission speed, W, yields packet size, multiplying both sides of the above equation by W establishes the average number of bytes that can be transmitted by an average node for each unit of time of the relative deadline. In a load-balanced network of n nodes, the capacity of the network is nU byte-hops per unit of relative deadline (or equivalently, nUr_{av} byte-meters per unit of relative deadline, where r_{av} is the average distance between communication hops). Hence, the real-time capacity of the sensor network, denoted C_{RT}, is bounded by

$$C_{RT} < nr_{\text{av}} \left(1 + \frac{\alpha}{N} - \sqrt{1 + \left(\frac{\alpha}{N}\right)^2}\right) W \tag{22.5}$$

Some interesting observations are apparent. First, rewriting $1 + \alpha/N$ as $\sqrt{1 + 2\alpha/N + (\alpha/N)^2}$, observe that when N is large, the term $(\alpha/N)^2$ can be neglected leading to

$$C_{RT} < nr_{\text{av}} \left(\sqrt{1 + \frac{2\alpha}{N}} - 1\right) W \tag{22.6}$$

We know from series expansion that for a small x, the term $\sqrt{1+x}$ is approximately equal to $1 + x/2$. Hence, substituting for the square root in Equation 22.6 when N is large, we get

$$C_{RT} < \frac{nr_{\text{av}}\alpha}{N} W \tag{22.7}$$

The above expression is the first known bound that establishes real-time capacity limits as a function of network size and node radio transmission speed. The bound errs on the safe side. It may be possible to communicate more traffic in a timely manner than what is predicted above.

22.6 Concluding Remarks

In this chapter, some basic principles for providing real-time wireless communication were addressed followed by an overview of real-time wireless protocols able to provide soft real-time guarantees. Finally, the notion of real-time capacity was introduced to quantify the capacity of a sensor network to deliver

information by deadlines. It is worth noting that while several protocols are available to support single-hop real-time wireless communication, the case of *ad hoc* real-time multihop still represents a challenging task and an open research problem the real-time and network communities are looking at. To the best of our knowledge, SPEED and RAP are the first delay-sensitive protocols devised for sensor networks, but they are still subject to unpredictable delays introduced at the MAC layer (due to the random backoff). Some interesting works exist that make the assumption of having globally synchronized nodes (see Refs. 8 and 25); they can avoid packet conflicts on the wireless medium by relying on a static and slotted packet schedule (TDMA), but they do not handle efficiently mobility or joining/leaving of nodes.

References

1. I. Aad and C. Castelluccia. Differentiation mechanisms for IEEE 802.11. In *IEEE Infocom*, Anchorage, Alaska, April 2001.
2. T. Abdelzaher, S. Prabh, and R. Kiran. On real-time capacity of multihop wireless sensor networks. In *IEEE Real-Time Systems Symposium*, Lisbon, Portugal, December 2004.
3. T. Abdelzaher, J. Stankovic, S. Son, B. Blum, T. He, A. Wood, and C. Lu. A communication architecture and programming abstractions for real-time sensor networks. In *Workshop on Data Distribution for Real-Time Systems*, Providence, Rhode Island, May 2003.
4. T. Abdelzaher, G. Thaker, and P. Lardiery. A feasible region for meeting aperiodic end-to-end deadlines in resource pipelines. In *IEEE International Conference on Distributed Computing Systems*, Tokyo, Japan, December 2004.
5. T. P. Baker. Stack-based scheduling of real-time processes. *The Journal of Real-Time Systems*, 3(1): 67–100, 1991.
6. R. O. Baldwin, I. V. Nathaniel, J. Davis, and S. F. Midkiff. A real-time medium access control protocol for ad hoc wireless local area networks. *SIGMOBILE Mobile Computer Communication Review*, 3(2): 20–27, 1999.
7. B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, J. Stankovic, and S. Son. An entity maintenance and connection service for sensor networks. In *The 1st International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.
8. M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *IEEE RTSS*, December 2002.
9. T. L. Crenshaw, A. Tirumala, S. Hoke, and M. Caccamo. A robust implicit access protocol for real-time wireless collaboration. In *IEEE Proceedings of the ECRTS*, July 2005.
10. R. Cunningham and V. Cahill. Time bounded medium access control for ad hoc networks. In *Principles of Mobile Computing*, 2002.
11. J. Deng and R. S. Chang. A priority scheme for IEEE 802.11 DCF access method. *IEICE Transactions on Communications*, E82-B(1): 96–102, 1999.
12. T. He, J. Stankovic, C. Lu, and T. Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. In *International Conference on Distributed Computing Systems*, Providence, Rhode Island, May 2003.
13. S. Jang-Ping, L. Chi-Hsun, W. Shih-Lin, and T. Yu-Chee. A priority MAC protocol to support real-time traffic in ad hoc networks. *Wireless Networking*, 10(1): 61–69, 2004.
14. S. Kumar, V. S. Raghavan, and J. Deng. Medium access control protocols for ad-hoc wireless networks: A survey. *Elsevier Ad-Hoc Networks Journal*, 4(3): 326–358, May 2006.
15. J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *ACM MobiCom*, 2001.
16. C. H. Lin. *A multihop adaptive mobile multimedia network: Architecture and protocols*. PhD thesis, University of California at Los Angeles, 1996.
17. C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7): 1265–1275, 1997.

18. C. R. Lin and M. Gerla. Real-time support in multihop wireless networks. *Wireless Networks*, 5(2): 125–135, March 1999.
19. C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real time environment. *Journal of the ACM*, 20(1): 40–61, 1973.
20. J. W. S. Liu. *Real-Time Systems*. Prentice-Hall, New Jersey, USA, 2000.
21. C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *Real-Time Technology and Applications Symposium*, San Jose, CA, September 2002.
22. R. Mangharam and R. Rajkumar. Max: A maximal transmission concurrency MAC for wireless networks with regular structure. In *IEEE Broadnets*, October 2006.
23. J. P. Monks, V. Bharghavan, and W. Hwu. A power controlled multiple access protocol for wireless packet networks. In *Proceedings of IEEE Infocom*, 2001.
24. A. Pal, A. Dogan, and F. Ozguner. MAC layer protocols for real-time traffic in ad-hoc wireless networks. In *ICPP*, 2002.
25. A. Rowe, R. Mangharam, and R. Rajkumar. Rt-link: A time-synchronized link protocol for energy-constrained multi-hop wireless networks. In *IEEE SEACON*, September 2006.
26. J. Sobrinho and A. Krishnakumar. Real-time traffic over the IEEE 802.11 medium access control layer. *Bell Labs Technical Journal*, 1(2): 172–187, 1996.
27. J. Sobrinho and A. Krishnakumar. Quality-of-service in ad hoc carrier sense multiple access networks. *IEEE Journal on Selected Areas in Communications*, 17(8): 1353–1368, August 1999.
28. F. A. Tobagi and L. Kleinrock. Packet switching in radio channels: Part II—the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *IEEE Transactions on Communications*, 23(12): 1417–1433, 1975.
29. N. Vaidya, N. Malpani, and J. L. Welch. Leader election algorithms in mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, August 2000.
30. A. J. Viterbi. *CDMA: Principles of Spread Spectrum Communication*. Prentice-Hall, April 1995.
31. Q. Wang, X. Liu, W. Chen, W. He, and M. Caccamo. Building robust wireless LAN for industrial control with DSSS-CDMA cellphone network paradigm. In *IEEE RTSS*, December 2005.
32. X. Yang and N. Vaidya. Priority scheduling in wireless ad hoc networks. In *ACM MobiHoc*, 2002.
33. J. Zatopiański, T. Jurdzińkski, and M. Kutylowski. Efficient algorithms for leader election in radio networks. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing*, July 2002.