Real-Time Communication and Coordination in Wireless Embedded Sensor Networks

SYSC 5701 – Operating System Methods for Real-Time Applications

Paul Boone
April 2004

Executive Summary

The goal of this report is to provide the reader with an understanding of the main issues in designing real-time communication solutions in wireless sensor networks. We begin by giving a brief overview of sensor networks, sensor limitations, and possible applications. In sensor networks, often a node's geographic location is more important than its global identifier. Messages often will need to be sent to a physical location. Communication protocols developed should be based on location-addresses as opposed to some global node identifier.

The main focus of this report is to communicate various ideas about what is required to provide guarantees that messages in the network are delivered according to their real-time deadlines. Some mechanism is required to assign priorities to packets that are to be sent on the sensor network. Three different strategies are discussed. The first solution is a MAC layer solution based on the IEEE 802.11 standard. In their solution, the authors arrange sensor nodes in a cellular structure and perform Earliest Deadline First (EDF) analysis to build deterministic contention free access to the underlying wireless medium. The second solution is a network layer solution. SPEED is a routing protocol designed to guarantee that messages move across the network (from source to target) at a predetermined speed providing soft real-time service. Finally, a communication architecture is described where multiple network layers work together to provide real-time packet prioritization and scheduling. The authors describe RAP, a communication architecture that introduces the novel idea of Velocity Monotonic Scheduling to prioritize packets at the network level based on their required velocity, as well as an enhanced IEEE 802.11 MAC layer to that is priority aware.

We learn from these strategies that in order to provide communication to meet real-time requirements in a scaleable fashion, a cooperative multi-layer architecture will be required. In addition, solutions must take into account both the time constraints as well as the physical distances between from the target nodes in order to schedule packets.

An additional part of this report was to introduce some selected sensor network projects. The TinyOS development environment is discussed, and a few other current projects in academia.

Open research areas include continuing research in cooperative multi-layer real-time communication architectures, wireless network security, improvements upon SPEED and RAP, and finally this author wishes to design and implement a real-time communications architecture using TinyOS and Berkeley motes.

Table of Contents

Executive Summary 2

1. Introduction 4

2. Scope 5

3. Current Research 6

3.1 MAC Layer Communication in a Cellular Structure 7
3.1.1 Intra-cell communication 8
3.1.2 Inter-cell Communication 9
3.1.3 Implicit contention using EDF with FRAme SHaring (FRASH) 9
3.1.4 Analysis 11
3.1.5 Experimental Results 11
3.1.6 Summary 12

3.2 A Stateless Protocol for Real-Time Communication in Sensor Networks 12
3.2.1 Design Goals 12
3.2.2 Implementation Details 14
3.2.3 Simulation and Evaluation 20
3.2.4 Summary 20

3.3 Real-Time Distance-Aware Scheduling Architecture - RAP 21
3.3.2 Location Addressed Protocol 23
3.3.3 Geographic Forwarding 23
3.3.4 Velocity Monotonic Scheduling 23
3.3.5 Priority Queues 24
3.3.6 MAC Layer Prioritization 24
3.3.7 Simulation and Evaluation 24
3.3.8 Summary 24

4. Selected Sensor Network Projects 25

4.1 TinyOS at Berkeley 25

4.2 S-MAC: An Energy Efficient MAC Layer 27

4.3 RoboMote Project 27

5. Open Research Areas 28

5.1 Cooperation of real-time communication protocol stack layers 28

5.2 Security 28

5.3 Improving the SPEED Protocol 28

5.4 Building a Sensor Network 29

6. Conclusion 29

Bibliography 30

1. Introduction

Wireless sensor networks are distributed computing systems that have many limitations. These include limited computing power, low memory, typically can run on batteries, and have low bandwidth available for communication.

A typical application for a sensor network (see Figure 1) could be a deployment of sensors to observe a physical area and report of any intruders entering the protected zone. The "x" marks a location where the sensor with the circle around it has detected an intruder of some kind. This information must be delivered to a base station of some sort in a timely fashion.

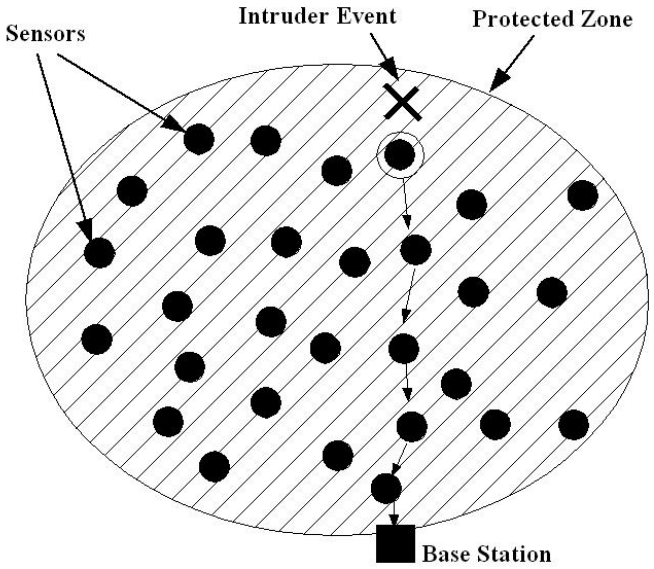


Figure 1 – Example Sensor Network

Sensor networks are very *data-centric,* meaning that the information that they collect about their environment must be delivered in a timely fashion to a collecting agent or base station. The data is typically collected externally to the network. Sensor networks are also tailored to their application. They are deployed to perform a specific task, and individual nodes are able to respond to stimulus in their environment in the way described for a particular application. Due to their unique characteristics, sensor networks are often treated as a special type of ad-hoc network.

Some issues that must be dealt with in developing a wireless sensor network are real-time constraints (messages arrive with bounded delays), resource constraints (power, bandwidth), unpredictability/unreliability of sensor nodes, and sensor density for the area of coverage.

In sensor networks, often a node's location is more important than its ID. This is especially true in very large networks, where this would consume valuable memory resources to keep track of global node IDs. Also, in sensor networks that are tracking information, the base station wants to know *where* something is happening and is not concerned with *which* sensor in particular is reporting.

In this report we discuss some current work done in developing real-time communication protocols and architectures for wireless sensor networks.

2. Scope

The paper will discuss current research on the development of protocols and architectures for communication and coordination amongst sensors in a real-time environment. The area of research is focused on the assignment of priorities and scheduling packets for transmission at both the *network* and *MAC* layers. As a secondary objective, we also discuss some selected sensor network projects that are currently on the go. Some of these projects are using the TinyOS environment to simulate and implement applications of sensor networks.

This report is divided into three parts. In the first part, we discuss some current research in the field of real-time communication in sensor networks. This is the main focus of the report. We examine the idea of a paradigm shift for designing sensor networks and some protocols that have been developed to ensure real-time network communication in sensor networks. In the second part we briefly introduce the TinyOS development environment as well as a few sensor network projects, some of which have been developed in the TinyOS environment. Finally, we discuss some open areas of research in real-time communication in sensor networks. These areas include the increased cooperation of the

various layers of the communication protocol stack to provide real-time services, as well as the aspect of wireless network security.

3. Current Research

In the following sections, we describe a few recent works in the area of providing real-time communication services in sensor networks. A lot of work has been done in the are of improving various routing protocols, or MAC layer protocols, but most of the work has been done in isolation (at a single specific layer). Improvements include increasing packet throughput, minimizing delay, or conserving energy. These are all valid goals at improving sensor network efficiency, but more needs to be done.

New research in communications in sensor networks examines all layers of the protocol stack. This is due to the fact that the network must interact with its physical environment (distance between sending nodes and target nodes).

At the MAC layer, protocols need to be designed which can enforce the message priorities based upon time and distance concerns depending on the physical environment. The network layer must also be aware of the physical environment. A node's *location* should be a key attribute to identify *addressable network objects* [12]. There have been some location-base routing algorithms, and location services, developed for ad-hoc networks, however these do not go far enough to enable addressing of network devices simply by their physical location.

In the first section, we describe a MAC layer solution for providing real-time communications guaranteeing that messages will meet their deadlines based on Earliest Deadline First (EDF) analysis.

In the second section, we described a real-time routing algorithm called SPEED that guarantees that packets will be delivered across the network at a minimum relative

velocity from source to target. SPEED is a simple routing protocol that requires little cooperation from the underlying MAC layer.

Finally, we describe RAP, which is a real-time communication architecture for large scale sensor networks. RAP provides query and event services for sensor networks, as well as a location-addressed transport protocol. The authors introduce a novel approach for scheduling (prioritizing) packets at a node based on Velocity Monotonic Scheduling (VMS), where a packet's required velocity is used to assign priorities. RAP also provides changes to the IEEE 802.11 protocol that implements priorities at the 802.11 MAC layer.

3.1 MAC Layer Communication in a Cellular Structure

In their paper [2], the authors describe a network architecture based upon a *cellular structure* [15] and the IEEE 802.11 standards. The IEEE 802.11 interfaces operate both in ad-hoc and infrastructure modes. They observe that current wireless medium access control (MAC) protocols are not up to handle traffic in a sensor network due to the fact that there are several differences from traditional ad-hoc networks. Messages in a sensor network are mainly periodic in nature, and need guaranteed bounded delivery delays. Sensors in the network often repeatedly collect the same information generating a lot of redundant data. Any MAC layer solution should try and optimize traffic flow to non-redundant data throughput meeting timing constraints versus simply maximizing data traffic flow throughout. The solution is base on the assumption that in most cases, the sensors in a network are fixed and they can be tracking targets moving within the network area.

The authors describe a MAC layer solution as a cellular network structure that exploits the periodic nature of data traffic to provide guaranteed bounded delays, while making use of all the available bandwidth of the wireless medium. They achieve their goal by scheduling real-time messages according to Earliest Deadline First (EDF) [14]. The authors assume that time is divided into frames, and all nodes are synchronized on a

frame basis. A single packet can be sent during a frame and all packets are a constant size.

3.1.1 Intra-cell communication

Real-time delay guarantee cannot be given unless a deterministic contention free schedule can be developed. In order to provide deterministic access to a wireless medium, it typically requires a priority encoding which takes considerable overhead. Because sensor networks have an advantage that most real-time traffic is periodic, and the underlying channels are of a multi-cast (broadcast) variety, this can be used to aid in the design of a MAC protocol based on EDF. The basis of the protocol is that the EDF schedule is duplicated at all nodes in the cell for packet transmission. If the schedules are the same, all nodes will know which node has the message with the earliest deadline, and access to the medium next. As an example, suppose we have three nodes *A, B,* and *C,* and all are given the message table as shown in Figure 2. The same schedule below the table is determined by each node using EDF (ties broken with node rank).

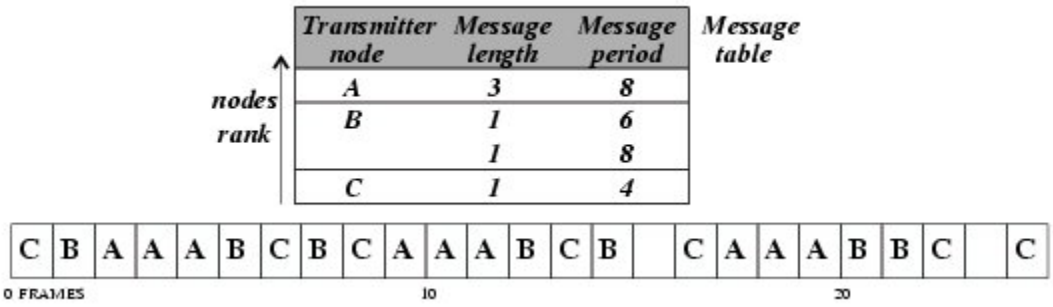


Figure 2 - Implicit Contention with EDF

Nodes can determine which frame they should send by simply counting frames and assuming all previous messages used their corresponding frames. If a node is listening to the channel and realizes that a previous message has completed early, it can use the FRAme SHaring (FRASH) algorithm that the authors define.

Also, in addition to frames reserved for intra-cell messages, some frames need to be reserved for inter-cell communication. This is determined a priori and is known to all nodes. These inter-cell frames must be synchronized among the nodes in the cell and must be calculated in the scheduling analysis as a blocking factor. If an inter-cell frame is reserved every T_{block} frames (T_{block} >= 2), the schedulability of intra-cell messages can be tested by [16]:

$$\forall i,\ 1 \le i \le n \quad \sum_{k=1}^{i} \frac{m_k}{T_k} + \frac{B_i}{T_i} \le 1,$$

assuming that messages are sorted by increasing relative deadline. The blocking of each message is the number of inter-cell frames that can occur during the message period:

$$B_i = \left\lceil \frac{T_i}{T_{block}} \right\rceil .$$

3.1.2 Inter-cell Communication

The authors define router nodes that are located at the center of cells to facilitate inter-cell communication. Routers have two transceivers so they can transmit and receive at same time on two different channels. Routers transmit on the channel of their cell and receive on the channel of the cell it expects to hear from at the given time (one of six adjacent cells). The inter-call messages are ordered by earliest deadline by each router node. During each inter-cell frame (synchronized across all cells) the router transmits and receives inter-cell messages according to a predetermined order of cell direction.

If the inter-cell communication mechanism is combined with a special routing protocol, end-to-end delay guarantee is given by summing the bounded delay at each router node in path along route (no routing protocol specified as this was out of scope of their research).

3.1.3 Implicit contention using EDF with FRAme SHaring (FRASH)

The authors describe their FRAme SHaring (FRASH) algorithm to aid in improving the throughput of traffic on the wireless channels while still guaranteeing delays. Due to the fact that each node has knowledge of the periodic intra-cell messages, a local EDF

scheduler can run in synch on each node. This means that nodes are aware of which messages are to be scheduled for transmission next. Also, any unused frames can be captured by using FRASH. These frames can be exploited via a simple aperiodic server. Whenever a transmission of a message does not use all its reserved frames, its identifier is placed in a FRASH field in the last data packet header of the current message. All nodes in the cell can detect this as no other node can transmit. As a result, the next eligible node can make use of the spare frames, or push them to the next eligible node.

An example given by the authors to illustrate the operation of their EDF and EDF-with-FRASH is as follows. They have a cell made up of three sensor nodes *A, B,* and *C* that are ranked alphabetically.

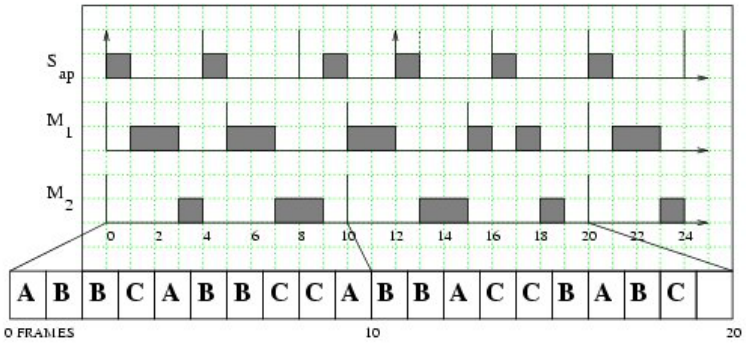


Figure 3 – Implicit EDF

The system has a set of periodic messages M_1 and M_2, of lengths $m_1 = 2$ and $m_2 = 3$. Their periods are $T_1 = 5$ and $T_2 = 10$. The sending nodes $N_1 = B$ and $N_2 = C$. In addition, node *A* also must send a message M_{ap} aperiodically. Its length is $m_{ap} = 4$ and has a minimum inter-arrival time $T_{ap} = 12$ with an average of *16*. A period server S_{ap} that has a capacity of 1 frame and period of 4 handles the aperiodic message, M_{ap}. Assuming M_{ap} first arrives at time = 0, and a second M_{ap} at time = 12, the EDF message schedule is shown in Figure 3 and is computed at all nodes.

Furthermore, by adding their implementation of FRASH, the schedule shown in Figure 4 can be achieved. In their small example we can see that by sharing frames it was possible to improve the handling of the aperiodic message M_{ap}.

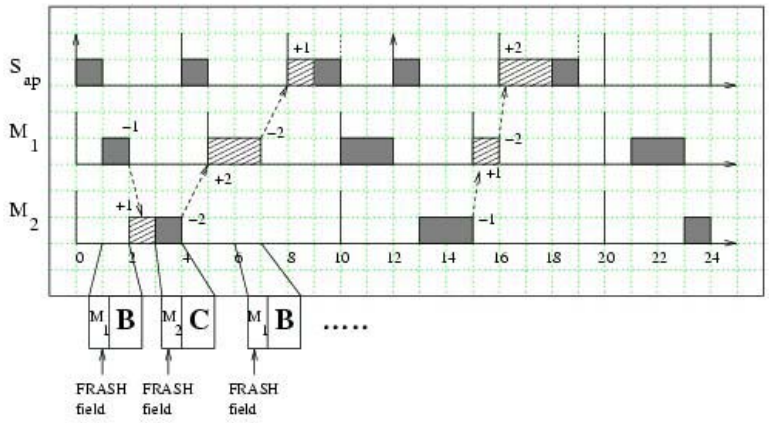


Figure 4 – Implicit EDF + FRASH

3.1.4 Analysis

The authors analyze the schedulability condition for a message set comprised of both hard periodic and soft aperiodic messages. By handling aperiodic tasked with a message server on every node, they behave like a periodic message scheduled by EDF. The main difference here is that messages can gain and yield extra frames from and to other messages. This run-time assignment of frames has no affect on the scheduling and therefore the message set can be guaranteed as schedulable by using Liu and Layland's condition:

$$\sum_{i=1}^{n} \frac{m_i}{T_i} + \sum_{j=1}^{s} \frac{Q_j}{T_j} \leq 1,$$

where m_i is periodic message length, T_i is the period, s in number of aperiodic servers, Q_j is server capacity and T_j is the server period.

3.1.5 Experimental Results

The authors developed a series of simulations in the ns-2 network simulator [29]. These included expanding the basic IEEE 802.11 CSMA/CD [17] protocol with their implicit EDF (I-EDF) and I-EDF with FRASH protocols. They measured two performance metrics: *system throughput* and *average delay* of aperiodic messages. They only tested their protocol with one cell (intra-cell communication). They compared their protocol

with basic CSMA/CD, EDCF [18] and Black-Burst [19]. For both performance metrics, I-EDF and I-EDF with FRASH greatly outperformed the other protocols as the number of nodes in a cell increased.

3.1.6 Summary

The authors have indeed provided a MAC layer protocol to provide real-time guarantees for message scheduling using Earliest Deadline First analysis. Their assumptions and setup are quite limiting however. They assume that the nodes are synchronized at the frame level. Also, packet size is fixed to be the size of one frame. To setup a sensor network using their cellular architecture, nodes need to be placed very specifically. Cellular network topology needs to be determined a priori and they only support fixed sensor networks (no mobility). Router cells must be placed at the center of cells in order to be able to communicate with all six neighbouring router cells. IEEE 802.11 channel assignment is complicated, as channels need to be assigned to cells in a way to reduce interference between cells. Experimentation has not been completed as the authors have only tested intra-cell communication. Dealing with inter-cell communication via the router nodes is a more complicated messaging scheduling problem.

3.2 A Stateless Protocol for Real-Time Communication in Sensor Networks

In their paper [4], the authors present a real-time communication protocol for sensor networks called SPEED. SPEED provides three real-time communication services, unicast, area-multicast and area-anycast. Unicast is a one-to-one communication from one sensor node to another. Area-multicast is a one-to-many communication where messages are sent from one node to all nodes within a specific area or region. Area-anycast is a communication from one node to at least one node within a specific area.

3.2.1 Design Goals

Unlike in a wired network where typically the delay is independent of the distance traveled from source to target, in a multi-hop wireless network, propagation delay is not

only dependant the distance from source to target, but also on the time spent at each node along the route from source to target.

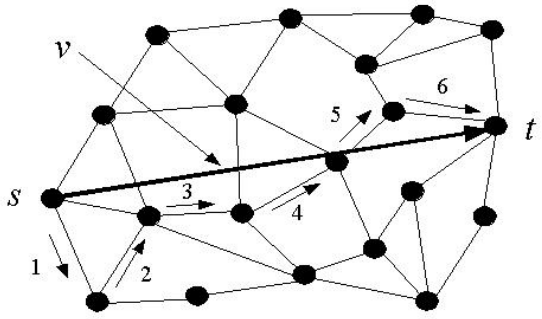


Figure 5 - SPEED: *v* is desired delivery speed across network

SPEED provides end-to-end *soft* real-time communication services between source and target nodes across a sensor network. This is achieved by maintaining communications at a desired delivery *speed*. This speed is the velocity traveled along a straight line from source to target. The protocol guarantees the delivery speed through a combination of greedy forwarding and feedback control on the network.

Figure 5 depicts the notion of SPEEDs *delivery speed* across the network. While the messages from source *s* to target *t* follows hops 1 through 6, the perceived velocity towards *t* will be the at least that of the variable *v*.

SPEED provides the communication services and follows the design objectives described below.

1. SPEED has a **stateless architecture**. Due to the physical limitations of sensor networks, including large scale, high failure rates, and small memory sizes, it is

becoming necessary to implement stateless protocols. SPEED only stores information for immediate neighbour nodes.

2. The protocol provides **soft real-time** services. SPEED provides a *uniform* delivery speed across a sensor network to meet the real-time demands of the desired application.
3. SPEED requires **minimal MAC layer support**. The protocol was developed with requiring the MAC layer to have real-time or quality of service (QoS) support. The feedback control mechanism allows for protocol operation with existing best-effort MAC layers.
4. Many reactive routing protocols developed for ad-hoc networks can adapt to avoid a hot spot or congested region in a network. However, most do not adapt well to a rapidly changing traffic patterns. SPEED introduces a backpressure re-routing mechanism to avoid large delays with minimal control messages to provide **QoS routing and Congestion Management**.
5. Due to the constrained bandwidth and energy resources in a wireless sensor network, it is often desirable to make use of multiple routes from source to target. The protocol implements **traffic load balancing** by using non-deterministic forwarding to balance flow among multiple routes.
6. SPEED is a **localized routing algorithm**, that is, all routing decisions are made independently at each node along the route from source to target. Actions of single nodes do not affect the entire network.
7. SPEED handles **void avoidance**. In greedy forwarding algorithms, sometimes there are no nodes closer to the target from the current node. When this happens, we hit a void. The protocol treats a void in the same way it handles congestion and guarantees that a greedy route will be found if one exists.

3.2.2 Implementation Details

SPEED does not use routing tables, but does make use of location information in order to route packets. It needs to know the location of its neighbours as well as the location of the target nodes. It is necessary that all nodes in the sensor network be **location-aware**.

The SPEED protocol is comprised of the following components.

1. SPEED API
2. Neighbourhood Beacon Exchange
3. Delay Estimation Scheme
4. Stateless Non-Deterministic Geographic Forwarding (SNGF) algorithm
5. Neighbourhood Feedback Loop (NFL)
6. Backpressure Rerouting
7. Last mile processing

3.2.2.1 SPEED API

The protocol provides four API calls: a) AreaMulticastSend, b) AreaAnycastSend, c) UnicastSend and d) SpeedReceived. For a) and b), input parameters include a **position** indicating coordinates in space, a **radius** which defines the distance from position, and a **packet**. For area-multicast, the packet is delivered to all nodes within the region defined by position and radius, and area-anycast delivers the packet to at least one node within the region. For c) unicast sending, a node **identifier** is required, along with a packet. Finally, d) provides a means for nodes to receive packets destined for them.

3.2.2.2 Neighbour Beacon Exchange

As most other location-aware routing protocols, SPEED uses periodic beaconing to exchange location information between neighbouring nodes. If nodes in the network are stationary or moving at a low rate, this beaconing can be reduced to a minimum. SPEED also introduces two additional beacons to be initiated on-demand. The first is a delay estimation beacon, and the second is a backpressure beacon. These additional beacons are used to quickly resolve changes of traffic patterns within the network.

3.2.2.3 Delay Estimation

The protocol uses *single hop delay* as the metric to estimate the load of nodes. Due to MAC layer handling of things, the protocol uses the unicast delay in determining which next hop neighbour is chosen. The metric is calculated without introducing new packets on the network, but by piggybacking the information on the data packets. The sender *timestamps* the packet when it places it in the network output queue. The receiver marks the acknowledgement packet with its *processing time*. When the sender receives the

ACK, it can calculate the single-trip time by subtracting the processing time from the round trip time. A current delay estimate is calculated by combining the new delay with previous results using the exponential weighted moving average (EWMA) [9].

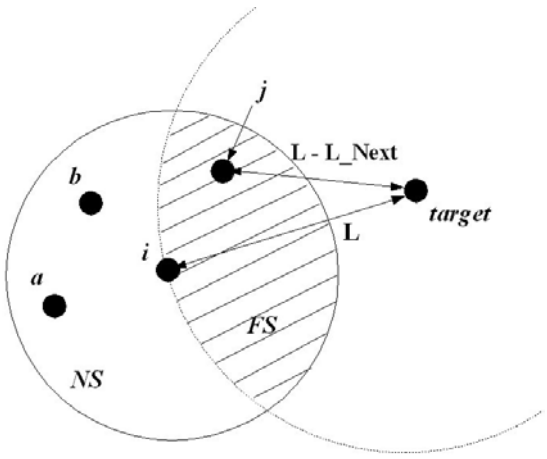


Figure 6 - Neighbour Set (NS) and Forwarding Candidate Set (FS) of node *i*.

3.2.2.4 Stateless Non-Deterministic Geographic Forwarding (SNGF)

The authors introduce three definitions before describing SNGF, refer to Figure 6. The first, *Neighbour Set* of node *i, (NS_i)* is the set of all nodes that are within range of node *i (a, b, and j)*. The second is the *Forwarding Candidate Set* of node *i*, (*FS_i(Destination)),* is the set of all nodes from the NS_i of *i* which are closer to the target *(simply node j in this case)*. Finally, *Relay Speed* is calculated by dividing the distance moved towards the target by the estimated hop delay. $Speed_i^j$(Destination) = $L - L_Next/HopDelay_i^j$. Since the protocol only keeps information on the neighbour set, and does not store routing tables, the memory required is only proportional to the number of neighbours.

The SNGF algorithm follows the following rules to determine where to forward packets.

1. Packets are only forwarded to nodes which are a member of the *FS_i(Destination)*. If no such node exists, the packet is dropped. A

backpressure beacon (will visit later) is sent to nodes in the reverse direction to prevent further dropped packets.

2. The protocol divides the nodes from 1. into two groups. The first group contains the nodes that have a relay speed greater than the desired speed, $S_{setpoint}$ and the second group contain the nodes which cannot maintain the required speed. ($S_{setpoint}$ is a system parameter)
3. The forwarding candidate is selected from the first group. The node that has the highest relay speed has the highest probability of selection for forwarding.
4. If there were no nodes that could meet the $S_{setpoint}$ speed parameter, a *relay ratio* is calculated based on *Neighbourhood Feedback Look* (next section).

SNGF has two properties that help achieve the design goals. First, since packets are forwarded to nodes that can maintain a desired forwarding speed, soft real-time end-to-end delivery is achieved with delay bound. *Delay Bound* = $L_{e2e} / S_{setpoint}$. L_{e2e} is the distance from source to target. $S_{setpoint}$ is the uniform speed that needs to be maintained across the sensor network. The second property is that SNGF can balance traffic and reduce congestion by sending packets out over a wide relay area.

3.2.2.5 Neighbourhood Feedback Loop (NFL)

The Neighbourhood Feedback Look (NFL) is an important part of the SPEED protocol used in maintaining the single hop relay speed above the desired $S_{setpoint}$ parameter.

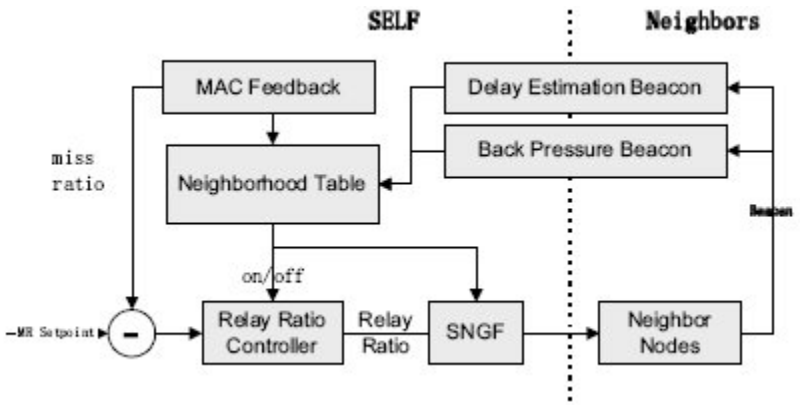


Figure 7 - Neighbourhood Feedback Loop (NFL)

Figure 7 shows the author's overview of NFL. The authors consider it a *miss* if a packet arrives at a node with a relay speed that is less than that of $S_{setpoint}$, or if a collision occurred. The percentage of these misses is the neighbours *miss ratio*. NFL's main goal is to reduce the miss ratios of the neighbours. The MAC layer gives feedback on the miss information to the Relay Ratio controller (RRC). The RRC will determine the relay ratio, which is fed back to SNGF. Based on the ratio, SNGF can make decisions on whether to drop or forward a packet.

3.2.2.6 Backpressure Rerouting

Backpressure rerouting can be achieved through collaboration with NFL and SNGF. Consider a case where there is a large volume of traffic flowing through part of the network. Some node, *i,* is trying to find a neighbour to forward a packet to, but all nodes in FS_i are congested (i.e. can not meet $S_{setpoint}$). In this case, the NFL is activated to help in backpressure rerouting. In node *i*, a certain percentage of packets will be dropped to ease the inflow of traffic in the congested region. At the same time, it also issues an on-demand *backpressure beacon*. This beacon includes its *ID*, the *Destination* and *AvgSendToDelay*. This delay is the average *SendToDelay* of all the nodes in $FS_{ID}(Destination)$. When a neighbour receives this beacon, it will see if the sender is in its FS(Destination). If it is, then the neighbour will update its SendToDelay according to the AvgSendToDelay value and can avoid sending packets to the congested region. If it is not in the FS(Destination) , then the neighbour ignores the backpressure beacon. This can reduce the chance of "false congestion indication" (traffic flowing out of congested region, or just outside the congested region, (through node *ID*) need not be affected).

An unfortunate side effect of this mechanism is that the backpressure beacon may cause a *chain reaction* of backpressure beacons (in the worst case where the whole network is congested) all the way back to the source, where it will be forced to stop sending packets to the target.

3.2.2.7 Void Avoidance

While greedy location-aware routing algorithms have many advantages over traditional routing algorithms (low overhead, tend to find shortest paths, no route discovery delay, no flooding), they do have some serious flaws. Often, a greedy forwarding algorithm may fail to find a path from source to target even if such a path does indeed exist. When a packet reaches a node that has an empty FS(Destination) set, it is said to have reached a *void*. SPEED handles rerouting around a void in the same way as it handles congestion. In Figure 8 we see an example of the void avoidance mechanism. As we can see, node 2 has no has an empty FS(*target*). It will then send a backpressure beacon with its ID, the Destination, and AvgSendToDelay = ∞ . Node 1 will set the SendToDelay for node 2 to ∞ and stop sending packets to node 2.

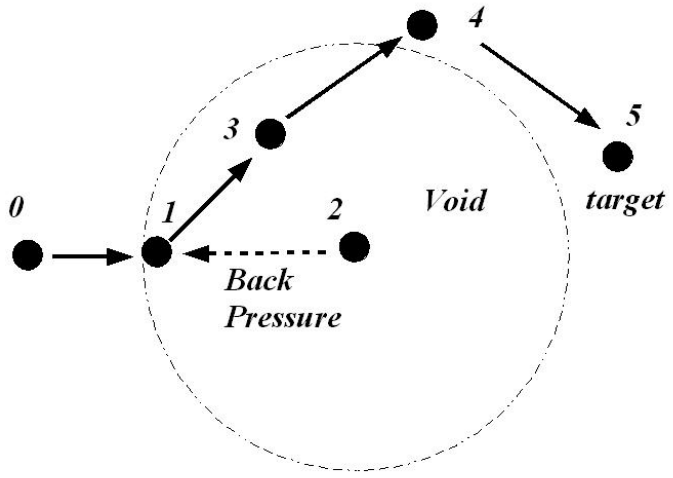


Figure 8 - SPEED Void Avoidance Scheme

3.2.2.8 Last Mile Processing

As SPEED is designed for sensor networks where IDs of nodes are not important, delivery is based upon a nodes geographic location. This is *last mile* processing since at this point the packet has left the control of SNGF algorithm. There are two services provided here: area-multicast and area-anycast. The delivery area is defined as a three dimensional point along with a radius. Nodes in the area can distinguish packet type based on type field. If the packet is an anycast one, the packet will be passed up the

protocol stack to the application and not forwarded. If it is a multicast packet, the first nodes to receive it in the delivery area set a time to live (TTL) field allowing the packet to live within the delivery region. Other nodes will keep a copy and rebroadcast it within the specified radius. Finally, unicast can be handled similarly to multicast, however, only the node with ID specified will pass the packet up the protocol stack.

3.2.3 Simulation and Evaluation

The authors performed a simulation of their SPEED algorithm in GloMoSim [11], as well as implementing it on Berkeley motes [10]. In their simulations, the authors compared SPEED with three other well-established ad-hoc routing algorithms including AODV [20], DSR [21] and GF [22]. Several performance metrics were measured including End-to-end delay under different congestion levels, packet miss ratio and void region avoidance. In end-to-end delay, SPEED improved upon the ad-hoc routing protocols by reducing the delay by between 30-40%. In the miss ratio simulation, with $S_{setpoint} = 1$ *km/s* determining a end-to-end delay of 200ms, SPEED outperformed the other algorithms as the packet rate increases. Finally, in studying the void avoidance, SPEED was only outperformed by DSR. This was expected as DSR floods the network in route discovery (does not deal with voids per se, assuming a connected network) and theoretically should find a path from source to target 100% of the time.

3.2.4 Summary

The authors present a routing protocol solution to guarantee real-time communication service in sensor networks. It does not work on the premise of packet deadlines, but by guaranteeing a minimal packet speed across the sensor network. The application is then required to make a decision on how to proceed. The protocol is highly scalable, as it is a localized algorithm, meaning that routing decisions can be made on a hop by hop basis without requiring global knowledge of the sensor network. The protocol can easily be implemented in the current environment of sensor networks. The authors claim of operation in with an existing underlying MAC protocol is misleading. They do not indicate how SNGF and NFL receive feedback from the underlying MAC layer. It seems that something must be added to the MAC layer in order for SPEED to operate. The

authors also implemented a stripped down version of SPEED (SPEED-S and SPEED-T which replaces SNGF with a MAX_SPEED (SPEED-S) and MIN_DELAY (SPEED-T) routing algorithms without backpressure) and few details were given.

It is unclear how the network can recover from backpressure messages, once a node has received one from a downstream node. Once the congestion in the area has subsided, how does an upstream node know that it is safe to send packets downstream again.

One problem with SPEED is that it does not guarantee packet delivery. Their void avoidance algorithm may result in dropped packets, but their experiments show this has been minimized. It is unclear of the authors' intent, but seems that if the avoidance scheme gets to a point where it drops a packet, the packet probably wasn't going to meet its "deadline" anyway.

Finally, it appears the value of $S_{setpoint}$ is fixed which does not allow for different classes of packets. The protocol guaranteed a fixed speed across the network for all application packets.

3.3 Real-Time Distance-Aware Scheduling Architecture - RAP

A new paradigm is required to aid in the development of large-scale real-time embedded sensor networks [1,3]. Traditional research in communication in sensor networks examines the need for meeting specific *time* constraints (deadlines). In their work, the authors introduce a protocol suite that not only deals with time constraints, but also takes into account the physical space constraints of sensor networks. By space constraints, they mean that the time required in sending a message from one sensor to another or to a base-station is dependent on the physical *distance* between the two points in space.

The authors develop a new protocol stack to address the problems of time and distance constraints. It implements a real-time message-scheduling algorithm that considers these

two factors. It also provides a transport-layer addressing that can associate a unique network address with external objects from the environment.

Messages in a real-time embedded sensor network must be delivered within a defined bounded time, or data received at target may be no longer relevant (stale). Also, sensor networks may be sending multiple messages of different urgencies to many different locations at various distances. The network protocols must be able to *order* the messages sent over the network to respect all time and distance requirements.

3.3.1 Overview

The authors have developed an architecture called RAP [5] to handle this task. RAP provides two contributions to the research area. To begin with, RAP is a real-time communication architecture for large-scale wireless sensor networks. It provides a set of high-level query and event services for applications. These services are based on a location-addressed or location-aware communication model provided by means of a lightweight network protocol stack. These services are not discussed as they are beyond the scope of this report.

The second contribution to advanced research is that RAP has incorporated the notion of a packet velocity, and implements *velocity monotonic scheduling* (VMS) as its default packet-forwarding scheme. Similar to SPEED, in order to meet the end-to-end latency bound, a packet must maintain some desired average speed or velocity across the network towards the target. This velocity is determined by the timed delay bound and the distance between source and target nodes.

RAP as an architecture, like SPEED is concerned about the physical geography of the network and distance plays a role in maintaining a desired speed or velocity across a sensor network. However, unlike SPEEDs SNGF routing algorithm, RAP does not provide a routing algorithm. Existing routing algorithms such as DSR [21] or GF [22] can be incorporated into the architecture.

3.3.2 Location Addressed Protocol

RAP includes the Location Addressed Protocol (LAP) that is very similar to User Datagram Protocol (UDP) with the exception that packets are addressed based on *location* instead of an ID like *IP address*. Like the SPEED protocol, LAP provides unicast, area-multicast and area-anycast communication.

3.3.3 Geographic Forwarding

As target nodes are identified by their geographical location, routing protocols must be aware of this. A node is aware of its location and can figure out the location of the target node relative to itself. In Geographic Forwarding (GF) [22], or greedy forwarding, chooses from among its neighbours the one that is closer to the target node. If no node is closer to the target, some mechanism must be invoked to route around the void. As GF used only local information (information about immediate neighbours) it is very scaleable. The authors claim that GF with location addressing can be used without a location directory service to save overhead.

3.3.4 Velocity Monotonic Scheduling

The most important part of a real-time communication architecture is how packets are scheduled for transmission on an outgoing link. Most routing protocols handle this in a first come first served (FCFS) basis. FCFS scheduling does not work well where packets have different deadlines for delivery. Packets should be prioritized based on their relative local importance. This should be done in both a deadline-aware and distance aware strategy. The RAP architecture orders the packets based on the velocity required. Analogously to rate-monotonic analysis, packets with a higher required velocity are given higher priorities than that of those with lower required velocities. Two variations of this theme were implemented. *Static velocity-monotonic* scheduling is where a packet's priority is determined at the source node based on delay and distance constraints and remains fixed for duration of transit. *Dynamic velocity-monotonic* scheduling allows for the protocol to vary the packet's priority during transit. If a packet encounters serious delays along its path, the priority can be increased. Similarly, the priority can be reduced if the packet is considered to be ahead of schedule.

3.3.5 Priority Queues

In order to manage this prioritization in a wireless network, priority queues were introduced at the nodes. Every packet is assigned a priority based on its requested velocity.

3.3.6 MAC Layer Prioritization

Local prioritization at each node is not sufficient to guarantee bounded delivery delays as sending nodes compete for a shared medium. The MAC layer needs to be modified to handle network contention. The authors adopted a schemed based on EDCF [13, 23], that has been introduced in the proposed IEEE 802.11e standard. In the authors' implementation, the 802.11 *DIFS counter* and *backoff window* were modified to make them priority-aware.

3.3.7 Simulation and Evaluation

The authors performed a simulation of their RAP architecture in GloMoSim. In their simulation they implemented GF, their two versions of VMS as well as the extensions to the IEEE 802.11 protocol. They compared the DSR and GF protocols while measuring the packet miss ratio. For packet scheduling, the compared their SVM and DVM algorithms against two baseline algorithms: FCFS and DS a deadline based algorithm.

Their results show that all deadline based scheduling algorithms performed better that the FCFS one. At the highest packet rate tested (over the largest distance), in GF with SVM, only 17.9% of packets missed their deadline as opposed to 77.6% for GF/FCFS and 46.0% for GF/DS. This result shows that SVM has an advantage by considering both deadline and distance in assigning packet priorities. The authors were surprised that DVM did not perform as well as SVM in their simulations.

3.3.8 Summary

With RAP, the authors present a scaleable real-time communication architecture for sensor networks. The RAP provides a suite of high-level query and event services, as well as a location-addressed transport layer. RAP provides a multi-layer communication

protocol stack that cooperates on prioritizing packets at not only the network layer, but also at the MAC layer. Their architecture allows the flexibility of incorporating any location-aware routing protocol desired. The authors introduce a novel approach to scheduling packets at the network level with VMS based on a packets requested velocity. It was surprising that DVM didn't perform as well as expected. The authors plan to further investigate DVMs performance and if it may be suitable for scheduling packets throughout a sensor network where different regions have different traffic congestion levels.

4. Selected Sensor Network Projects

The purpose of this section is to give a brief mention to some current research work on the implementation of real sensor networks in academia. Here we mention several projects, including the TinyOS development environment developed for motes available from UC Berkeley. Other project mentioned include those that were designed using TinyOS such as S-MAC, as well as other projects like the RoboMote project

4.1 TinyOS at Berkeley

TinyOS [24] is an open source operation system designed at UC Berkeley for wireless embedded sensor networks. TinyOS was originally designed for development of applications for Berkeley motes. TinyOS comes with a simulation environment where developers can test their sensor network applications on various network deployments. Figure 9 is included to show the TinyViz environment where developers may simulate and run their application code on multiple network topologies. While running TinyViz, developers can access such things as setting breakpoints, viewing debug messages for individual sensors, sent radio packets, and current radio links.

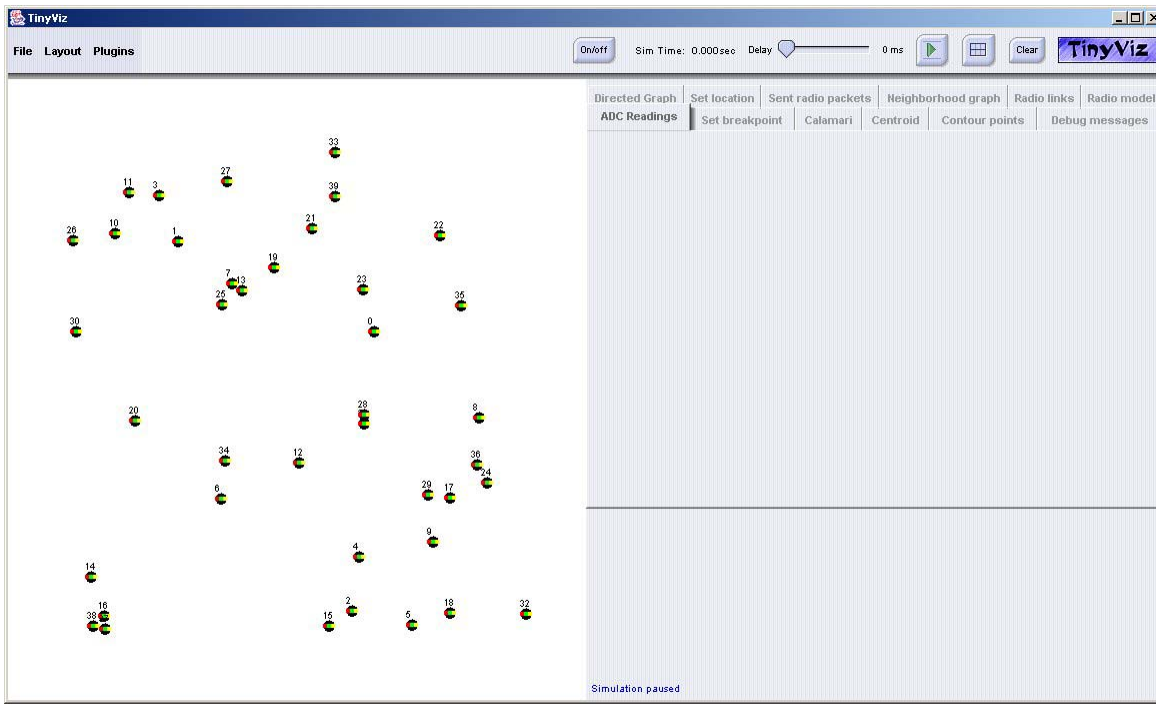


Figure 9 – TinyViz Screenshot

A complete install of the TinyOS environment for windows (including Cygwin, Java JDK, and the TinyOS environment) can be found here:

http://webs.cs.berkeley.edu/tos/windows-1_1_0.html

The following was as list of projects taken from the TinyOS related work web site [26] and lists several current projects currently ongoing at UC Berkeley**:**

Related UC Berkeley work

- Calamari *Localization solutions for sensor networks*
- CotsBots *Cheap, Off-the-Shelf Robots using TinyOS*
- Firebug *Wildfire instrumentation, a project of the Berkeley Civil Engineering project*
- Great Duck Island *"Our goal is to enable researchers anywhere in the world to engage in non-intrusive monitoring of sensitive wildlife and habitats. Sensor motes … are monitoring the nesting habitat of the Leach's Storm Petrel on the island and relaying their readings … into a satellite link that allows researchers to download real-time environmental data over the Internet."*
- PicoRadio *Ad hoc, Ultra low-power Wireless Networking*
- Sensing Structural Integrity *Sensors report the location and kinematics of damage during and after an earthquake.*
- Telegraph *A database customized for streaming data such as that found in sensor networks*

- TinyDB *A query processing system for extracting information from a network of TinyOS sensors.*
- TinyGALS *A programming model for event-driven embedded systems, an EECS project*
- XYZ On A Chip *Integrated Wireless Sensor Networks for the Control of the Indoor Environment in Buildings* [26]

4.2 S-MAC: An Energy Efficient MAC Layer

S-MAC (sensor-MAC) [28] is an energy-efficient MAC layer protocol designed for use in wireless sensor networks. It has been implemented in TinyOS and on Motes. Their MAC layer implementation tries to conserve energy by implementing a sleep/wakeup cycle that allows sensors to be sleeping most of the time. A current release of their software is available here: http://www.isi.edu/ilense/software/smac/.

4.3 RoboMote Project

RoboMote is a part of the Scalable Coordination of Wireless Robots (SCOWR) project at USC's Robotics Embedded Systems Lab. RoboMotes are small cheap robots [24]. Each robot has a wireless transmitter, two speed and direction controlled wheels (with optical encoders for odometers), a solar cell for continuous power, a compass for direction, and bump and infra-red sensors for obstacle avoidance.

The RoboMote team believes that the ability of sensor nodes to move, or influence its location will be important in the future of sensor networks. The ability of nodes in sensor networks to move around to best accomplish a task may be an invaluable feature.
Further information can be found at the following URL:
http://robotics.usc.edu/~robomote.

5. Open Research Areas

In this section we discuss four open research areas in the field of real-time communication and coordination of sensor networks: multi-layer protocol cooperation and wireless security, improving existing protocols (SPEED) and a sensor network implementation.

5.1 Cooperation of real-time communication protocol stack layers

Current research is required to investigate how to develop a cooperating multi-layer real-time communication protocol stack to guarantee message delivery meets the real-time deadlines. Much work has been done investigating individual layers, for example providing an improved routing protocol, or improving the MAC layer, but more needs to be done. As more large-scale real-time sensor networks are developed, a scaleable architecture including cooperating layers of the communication protocol stack will be needed to guarantee real-time requirements and reduce design and implementation costs.

5.2 Security

There is much work that needs to be done in the area of wireless network security, and this adversely affects the ability of sensor networks to operate in a real-time environment. Wireless sensor networks are susceptible to eavesdropping as well as various denial of service (DoS) attacks. For example, in the RAP protocol, a malicious node in a sensor network could exploit vulnerabilities in the protocol by introducing a large number of high velocity packets (flooding). This can be accomplished by marking the packets with either a short deadline or with a very long distance. Security is of great concern in all wireless networks, and in particular to sensor networks with real-time mission critical tasks.

5.3 Improving the SPEED Protocol

There are a few loose ends to deal with in the SPEED protocol. First, their void avoidance mechanism could be improved to eliminate dropped packets. Finally, it would be beneficial introduce different packet classes, which would allow for a source node to set a desired speed for its packet. The impact of these changes should be studied to see if the additional complexity is warranted.

5.4 Building a Sensor Network

From this authors perspective and interests, there is a plan to develop a mobile sensor network application that can enable sensor nodes to interact with their environment. This work will be completed in the TinyOS development environment.

6. Conclusion

In this report we have seen that there are many issues involved in developing real-time communication protocols for sensor networks. Sensor nodes have limited resources including computation power, communication bandwidth, and battery power.

Any communication protocol for sensor networks must ensure that messages are delivered with bounded delay to meet real-time deadlines. There are several different strategies to develop real-time communication and coordination in wireless sensor networks. We have seen three such strategies. The first was a MAC layer solution that provided packet scheduling based on Earliest Deadline First analysis to guarantee packets meet delivery deadlines. The second was the SPEED protocol that provided soft real-time services by guaranteeing packets traveled a minimum relative velocity across the network. The final strategy was a communication architecture that provided packet prioritization at both the network and MAC layers. With their SVM algorithm and an extended IEEE 802.11 MAC layer, the authors achieved significantly lower miss rations over the simple GF algorithm with FCFS and standard IEEE 802.11 MAC layer.

We learn from these strategies that in order to provide communication to meet real-time requirements in a scaleable fashion, a cooperative multi-layer communication architecture will be required. In addition, solutions must take into account both the time constraints as well as the physical distances between from the target nodes in order to schedule packets.

Open research areas include continuing research in cooperative multi-layer real-time communication architectures, wireless network security, improvements upon SPEED and RAP, and finally this author wishes to design and implement a real-time communications architecture using TinyOS and Berkeley motes allowing mobile sensor nodes to interact with their environment.

Bibliography

[1] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, J.C. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks", *Proceedings of the IEEE*, vol. 91, num 7, pp. 1002-1022, July 2003

[2] M.Caccamo, L.Y. Zhang, L. Sha and G. Buttazzo, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks", *IEEE Real-Time Systems Symposium (RTSS '02)*, 2002

[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 Embedded Sensor Networks", *International Conference on Distributed Computing Systems Workshops*, 2003

[4] T. He, J. Stankovic, C. Lu and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communications in Sensor Networks", *International Conference on Distributed Computing Systems (ICDCS'03)*, 2003

[5] C. Lu, B. M. Blum, T. F. Abdelzaher, J. Stankovic, T. He, "RAP: A Real-Time Communications Architecture for Large-Scale Wireless Sensor Networks", In *Real-Time Technology and Applications Symposium*, San Jose, CA, October 2002.

[6] Y. Li, W. Ye, and J. Heidemann, "Demonstration of Schedule and Latency Control in S-MAC", *USC/ISI Technical Report ISI-TR-581*, University of Southern California, November 2003.

[7] D. Gay, M. Welsh, P. Levis, E. Brewer, R. von Behren, D. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A Survey on Sensor Networks", *IEEE Communications Magazine*, pp. 102-114, August 2002

[9] J. F. Kurose, K. W. Ross. Computer Networking A Top-Down Approach Featuring the internet. ISBN 0-201-47711-4 Addison Wesley Longman Inc. 2000

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Network Sensors. *ASPLOS 2000*.

[11] X. Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation PADS '98*, May 26-29, 1998.

[12] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8), August 2001.

[13] I. Aad and C. Castelluccia. Differentiation mechanisms for IEEE 802.11. In *IEEE Infocom*, Anchorage, Alaska, April 2001.

[14] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in hard real time environment. Journal of the ACM, 20(1):40-61, 1973.

[15] D. J. Goodman. Cellular packet communications. IEEE 7ransactions on Communications, 38(8), August 1990.

[16] T. P. Baker. Stack-based scheduling of real-time processes. The Journal of Real-Time Systems, 3(1):67-100, 1991

[17] L. Kleinrock and F. Tobagi. Packet Switching in radio channels: Part I – carrier sense multiple-access modes and their throughput delay characteristics. IEEE Trans. Commun., COM23(12):1400-1416, 1975

[18] J. Sobrino 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, 1999

[19] M. Benveniste, G. Chessno, M. Hoeben, A. Singla, H. Teunissen, M. Wentink. EDCF Proposed Draft. IEEE Working Document 802.11/131r1, 2001

[20] C. E. Perkins and E. M. Royer. Ad-hoc On Demand Distance Vector Routing. In *WMCSA'99*, February 1999.

[21] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[22] I. Stojmenovic and X. Lin. GEDIR: Loop-Free Location Based Routing in Wireless Networks, *IASTED Int. Conf. on Parallel and Distributed Computing and Systems*, November 3-6, 1999.

[23] Mangold, S. and Choi, S. and May, P. and Klein, O. and Hiertz, G. and Stibor, L. IEEE 802.11e Wireless LAN for Quality of Service (invited paper). In *Proceedings of the European Wireless*, Vol. 1, pp. 32-39, Florence, Italy,

[24] Gabriel T. Sibley, Mohammad H. Rahimi and Gaurav S. Sukhatme, "Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks"(pdf) *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2002)*, 2002 (http://robotics.usc.edu/~robomote/)

[25] TinyOS Website: http://webs.cs.berkeley.edu/tos/

[26] TinyOS related work website: http://webs.cs.berkeley.edu/tos/related.html

[27] sensor-MAC (S-MAC) an energy efficient MAC layer for sensor networks. http://www.isi.edu/ilense/software/smac/

[28] W. Ye and J. Heideman, Medium Access Control in Wireless Sensor Networks, USC/ISI Technical Report ISI-TR-580, October 2003.

[29] The Network Simulator - ns-2. **http://www.isi.edu/nsnam/ns/ns-documentation.html**