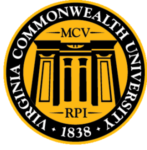


Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations | Graduate School

2007

Supporting Data-Intensive Wireless Sensor Applications using Smart Data Fragmentation and Buffer Management

Mbonisi Masilela
Virginia Commonwealth University

Follow this and additional works at: https://scholarscompass.vcu.edu/etd

Part of the Computer Sciences Commons

© The Author

Downloaded from
https://scholarscompass.vcu.edu/etd/779

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

المنارة للاستشارات

www.manaraa.com

Supporting Data-Intensive Wireless Sensor Applications using Smart Data Fragmentation and Buffer Management

A thesis submitted in partial fulfillment of the requirements for the degree Master of Science at Virginia Commonwealth University

By

Mbonisi Masilela

Director: Dr. Ju Wang
Assistant Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

December 2007

المنارة للاستشارات

www.manaraa.com

Acknowledgements

First and foremost, I would like to thank Dr. Wang for being such a great advisor and introducing me to this area of Wireless Communications. It has always been a dream of mine to work on, design and program wireless devices and under the guidance of Dr. Wang this dream has been realized.

I would also like to thank Dr. Choi, Dr. Hughes and Dr. Rafiq for introducing me into the area of embedded systems and awarding me the opportunity to learn and perfect my skills in this area. In addition, they have been great mentors that have helped me grow academically and intellectually.

In addition, I would like to thank Deanna Pace along with the faculty of the Computer Science Department for facilitating the execellent academic experience that I have had here at Virginia Commonwealth University.

Finally I would like to thank my friends and family for the great support that they have provided me through the course of my studies.

المنارة للاستشارات

www.manaraa.com

Table of Contents

List of Figures iv
Abstract v
1. Introduction 6
2. Lightweight Data Transportation Protocol 11
   2.1 Problem Description and Packet Structure 11
   2.2 Protocol Behavior 14
   2.2.1 Session Initiation Phase 14
   2.2.2 Data Transfer Phase 16
   2.2.3 Session Termination Phase 16
   2.3 Error Recovery 17
   2.4 Timeout Handling 18
3. Pipelined Transmission 19
   3.1 Sensor nodes synchronization and TDM based Transmission 20
4. Retransmission Policy and Buffer Management 24
   4.1 Retransmission Policy 24
   4.2 Buffer Management Policy 25
5. Hardware and Software Platform Details 26
6. Experiment Setup 27
   6.1 Test Setup Instructions 27
   6.2 Implementation Details 32
   6.3 Source Code Outline 35
7. Protocol Evaluation 37
8. Conclusion and Future Work 39
List of References 40
Appendix A – Source Code 43

المنارة للاستشارات
www.manaraa.com

List of Figures

Figure 1: (a) Mica2 and (b) VCU Wireless Sensor 7
Figure 2: Packet Structure 13
Figure 3: Transmission Sequence Overview 14
Fig. 4: FS scheduling pattern for 6 nodes. 20
Table 1: Synchronized local time at sensor nodes 22
Figure 5: Distributed Buffer Overview [17] 25
Figure 6: AVR Studio Window 28
Figure 7: AVR Studio Device Programming Window 29
Figure 8: Test file location – X:\Code\Board Test\cmote_test 30
Figure 9: Schematic and PCB file location – X:\Design Projects\zmote_2.0\logic and layout 31
Figure 10: Gerber File location – X:\Design Projects\zmote_2.0\CAM 32
Figure 11: Sensor PCB Layout View 32
Figure 12: Session Capture from the Snooper Node 34
Figure 13: Sample Successfully Transported Image 34
Table 2: Lightweight Data Transportation Protocol Results: Data Rate vs. Retransmission Rate 37
Table 3: Stateless Fragmentation Results: Data Rate vs. Retransmission Rate 38

www.manaraa.com

Abstract

Supporting Data-Intensive Wireless Sensor Applications using Smart Data Fragmentation and Buffer Management

A thesis submitted in partial fulfillment of the requirements for the degree Master of Science at Virginia Commonwealth University

Virginia Commonwealth University, 2007

Director: Dr. Ju Wang, Assistant Professor, Department of Computer Science

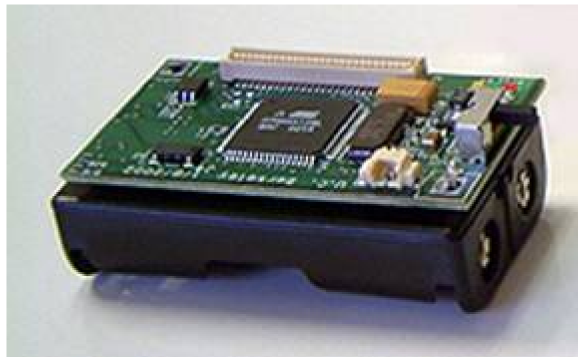
Recent advances in low power device technology have led to the development of smaller powerful sensors geared for use in Wireless Sensor Networks. Some of these sensors are capable of producing large data packets in a single reading. This becomes a challenging problem given the constraints imposed by current MAC and Transport Layer implementations since a single data packet can exceed the MTU of the protocol stack. Little has been done in the way of addressing this issue in Wireless Sensor Networks. This paper proposes a novel solution to this issue. Proposed is a Lightweight Data Transportation Protocol that uses smart data fragmentation and efficient pipelined transmission and buffer management schemes to solve this problem. The methodology outlined in this paper ensures that data is successfully transmitted from source to destination with minimal delay or packet loss.

المنارة للاستشارات

www.manaraa.com

1. Introduction

Wireless Sensors Networks are small low power embedded systems geared towards monitoring applications. Typically they are composed of a microcontroller, a radio communications interface and interfaces for sensor input. They are available in a variety of configurations and Figure 1.a below shows an example of the Mica2 wireless sensor from Crossbow and Figure 1.b shows the VCU Mote that we designed. These sensors can be connected together to form large interconnected networks using various topologies such as star, cluster tree and mesh topologies. All these nodes on the network must eventually connect to a computer usually known as the sink.



المنارة للاستشارات
www.manaraa.com

Figure 1: (a) Mica2 and (b) VCU Wireless Sensor

The sink is usually the connection to the outside world and is typically connected to the internet. This allows for information obtained from the sensor networks to he transported to any point on the globe. In addition, internet connectivity allows us to control the sensor network if necessary. These sensors are usually battery powered therefore making energy conservation an important issue. They mostly go through sleep-wake cycles in order to reduce power consumption. Depending on the usage of these systems, they can last for years without any battery replacements. These sensors are available commercially and can be used for a variety of applications.

Programming wireless sensors can be done using a variety of technologies. An open-source operating system known as TinyOS is the most widely used platform for programming these devices. The software is written in a language called NesC. This

المنارة للاستشارات
www.manaraa.com

language was designed to avoid some of the pitfalls of programming in C/C++, although these sensors can still be programmed using C/C++.

Typically they are used for environmental monitoring where they are deployed in a targeted area for measuring temperature, humidity and light. They were originally intended to be low traffic and data rate systems. But as with most systems, their function has begun to evolve. For instance, Wireless Sensors Networks are now being used in Bio-Medical applications [1] and [2] to name a few, to transport critical physiological readings from patients, in what are known as Personal Patient Area Networks, as defined in [3]. Some of these Bio-Medical sensors produce large volumes of data per reading that otherwise exceed the MTU (Maximum Transmission Unit) of most MAC layers. In particular, devices such as the Pulse Oximeter produce large data packets of 125 bytes per reading at least 3 times per second. With the inclusion of other sensor data like EKG, Temperature, Galvanic Skin Resistance, CO_2 data and other readings, it is foreseeable that the final data packet would exceed the MTU as stated earlier.

Other applications that may generate large packets of data include the use of low resolution CMOS camera devices in sensor networks. These may be used to periodically take pictures for monitoring purposes such as in environment monitoring applications. CMOS cameras have the capability of producing images at a 640 x 480 resolution which may be 30 KB in size. In addition, the transportation of data logs recorded by sensors may also be voluminous since the logs may have been recorded over long periods of

المنارة للاستشارات
www.manaraa.com

time. In applications such as these, it is critical that delivery of the data is guaranteed and is delivered in a timely manner with minimal overhead. In addition, partial data is rendered useless since a complete data profile is required for correct interpretation and analysis.

Some application specific MAC layers have been developed for these types of applications such as [4] with respect to Bio-Medical applications, but none particularly address the issue highlighted above. ZigBee [5] supports a large MTU size of up to 127 bytes, and S-MAC supports an MTU of up to 250 bytes. However these MTU sizes are still smaller than the size of an image captured by a CMOS camera. Due to these constraints, one application level sensor reading is transported over several small packets that are subject to individual delay and transmission error. Any transmission error on one packet will affect the overall delivery at the application layer. It becomes essential for efficient transportation of such data. In the case of [2], it would be undesirable to transmit each sensor reading separately because given the number of sensors per complete reading, several packets would need to be transmitted resulting in poor bandwidth usage. [6] has shown that the transmission of fewer and larger packets yields better performance than the transportation of more smaller packets.

Whilst it is possible to implement large MTU support in the wireless sensor networks for a particular application, where the application can pack as much data into a single packet as needed, it may not always produce satisfactory network performance. Our proposed

www.manaraa.com

solution uses smart data fragmentation that is customized for wireless sensor networks. Data fragmentation is a widely used concept in many communications hardware and software, but little has been said in the context of Wireless Sensor Networks. Internet traffic is often fragmented at the transportation layer through the use of TCP [7] and could be further fragmented at the MAC layer as it travels through different network infrastructures. There have been efforts to port the protocol stack to Wireless Sensor Networks such as [8], however the TCP/IP protocol suite in itsself is too complicated a protocol for Wireless Sensor Networks. In addition, the protocol is very power-intensive which makes it undesirable to use in low power sensor networks.

We propose a novel application layer level protocol, Lightweight Data Transportation Protocol that incorporates the use of efficient Buffer Management techniques. Buffer management is critical since it provides for the smoother flow of data in case of a transmission problem due to congestion or node failure. In the rest of this thesis, I discuss the presented methodology in detail.

The source code provided with this document is written in NesC and Java. NesC was used to program the Mica2 sensors for the TinyOS platform. The PC based data processing software was written in Java.

www.manaraa.com

2. Lightweight Data Transportation Protocol

For Wireless Sensor Networks, most existing work concentrates on reliable transportation [9,10], MAC [5,11,12] and network [13],[14] layer design. However, there has been little work that addresses the transportation need for specific applications. The proposed network protocol is an application-oriented cross-layer design. We believe that application-specific requirements need to be considered in all layers for efficient communication. In particular, given the multi-hop and unreliable nature of communication links in wireless sensor networks, it is important to build fault-tolerant mechanisms for reliable end-to end transmission. In the following sections, we outline the protocol structure and signaling behavior for normal operations.

2.1 Problem Description and Packet Structure

We assume that the routing information has been provided and routing paths have been established in advance in accordance to the network architecture. We assume that each node can support a maximum of M connections, and each connection has N buffers allocated. In a typical scenario, a relaying node will route traffic for multiple connections.

As mentioned in the introduction, a major problem of transmitting large application-level data units through long-haul sensor networks is that a single packet drop might

المنارة للاستشارات
www.manaraa.com

significantly delay the delivery of application level data. Using conventional stateless fragmentation is insufficient.

Our solution is to design a state-full fragmentation scheme such that the intermediate relaying nodes are aware of the fragmentation information which can be used for buffer allocation and retransmission scheduling in the event of packet drops. For this purpose, we designed and implemented a new packet structure and a suite of signaling processes.

Figure 2 below illustrates our proposed packet structure with both the DATA field and management fields. The management fields only occupy 6 bytes with a 2 byte CRC field totaling 8 bytes. These fields serve multiple purposes during a transmission session so as to minimize the overhead of using extra fields. The DATA field contains the application layer payload. This field has a maximum value given by the formula below:

$$MaxSizedata = MTU - 8$$

These fields are used to ensure that the packet is reassembled in the correct order at the destination.

The FD field is the Packet Delimiter Field that indicates the beginning of a data packet for synchronization purposes. The PKTID field is a unique transaction identifier used to group packets that belong to the same fragmented data reading. This field remains constant throughout the transmission session. The SEQNO field is a packet sequence number that indicates the order in which packets should be reassembled once they arrive at their destination.

المنارة للاستشارات

www.manaraa.com

The PKTYPE field indicates the type and purpose of packet being transmitted. The supported packet types are: INIT, ACK, DATA, CLOSE, EXIT, RESEND and ABORT. Most of the packet types are self-explanatory. Here we briefly summarize the management type packets.

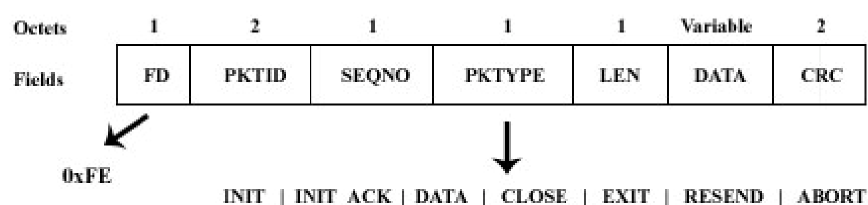


Figure 2: Packet Structure

INIT and INIT ACK indicate a transmission session initiation attempt and response. The INIT ACK response must be sent by the receiver as soon as the INIT request is received. Packet type CLOSE is issued by the sender to indicate the completion of transmission of all data fragments. EXIT packet is a receiver based response used to indicate that all packets were successfully received and the sender can terminate the data transfer session.

Upon the completion of packet accounting activities, if it is determined that there are missing or corrupt packets, a RESEND packet can be issued to retrieve the packets in

المنارة للاستشارات
www.manaraa.com

question. It can also be issued by a routing node if the packet being routed is deemed corrupt. We use 16 bit CRC-CCITT for error calculation in the CRC field.

2.2 Protocol Behavior

The complete data transfer algorithm can be divided into three distinct phases. These are the Session Initiation Phase, the Data Transfer Phase and the Session Termination Phase. Each phase is discussed in detailed in its respective section below. Figure 3 illustrates the Data Transmission Algorithm.

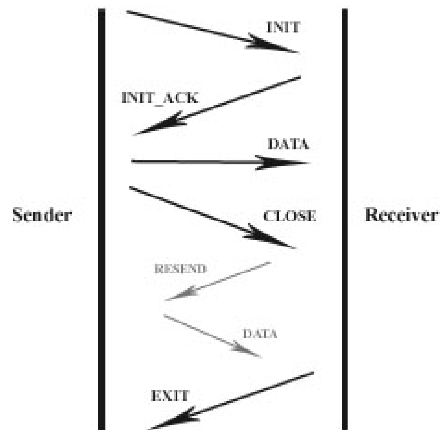


Figure 3: Transmission Sequence Overview

2.2.1 Session Initiation Phase

المنارة للاستشارات
www.manaraa.com

When a node becomes ready to transmit its data, it initiates communication with the destination node. The sender must first issue an INIT packet to the receiver. The SEQNO field should contain the total number of DATA packets that are going to be sent. The DATA portion of this message may contain some information about the data that is being sent. This could be MIME content type information, target file name, timestamp information, etc. The receiver should know the specific format and arrangement of this information otherwise it may be rendered unusable. Any routing nodes along the transmission path should always acknowledge the receipt of this packet to the previous hop and buffer the packet using the methodology defined in [15]. This buffering methodology is discussed in the next section.

Upon reception of the INIT packet, the receiver must immediately respond with an INIT ACK. In addition, the receiver must allocate the necessary resources to handle the incoming data from the sender. Every node that routes the INIT ACK packet must modify the SEQNO field by incrementing it by one. This will be used to indicate the number of hops from source to sink. In addition, the first 4 bytes of the data field will be used to determine the average propagation delay from hop to hop. Each hop that routes this packet must add its estimated next-hop propagation delay to the 4 byte data field. This propagation delay can be estimated using the Pair-Wise Time Synchronization algorithm defined in [16]. The average propagation delay will be used in the timeout handling phase of our data transmission algorithm.

المنارة للاستشارات
www.manaraa.com

When the sender receives the INIT ACK packet, it must examine the packet in order to extract the Estimated Total Delivery Time from the first 4 bytes of the data field. Once the information is acquired, data transmission can begin.

2.2.2 Data Transfer Phase

The sender must begin transmitting the fragmented data packets sequentially. The packets are transmitted in the order dictated by the sequence numbers. Each packet must be acknowledged by the next hop to ensure successful routing transmission. As with every packet, the CRC must be validated in order to ensure the validity of the data being transmitted.

2.2.3 Session Termination Phase

Once all the data packets have been transmitted, the sender must issue a CLOSE packet to signify that it has completed the transmission of all data packets. Upon reception of the CLOSE packet, the receiver must make sure that there were no lost packets during the Data Transmission phase. If any packets are found to be missing, a retransmission request is issued using the RESEND packet for the missing packet. The SEQNO field will be used to indicate a missing packet.

According to [15], routing nodes buffer all packets sent from the source to sink. When a routing node receives a retransmission packet, it must first search its primary buffer for the missing packet in question. If the packet is found, it is immediately sent to the

المنارة للاستشارات
www.manaraa.com

requester. If the routing node fails to locate it in its primary buffer, it must check its secondary buffer as well. This process takes nanoseconds and has little to no significant impact on the transmission time. If it fails to find the packet in its secondary buffer, it forwards the RESEND packet downstream to the next hop that in turn performs the same search sequence. If none of the routing nodes find the packet in their buffers, the RESEND request eventually makes it back to the source node. The source node should promptly respond to the request by resending the requested packet.

Once all packets have been retransmitted, the receiver must issue an EXIT packet to indicate to the sender that it has received all the packets. Along the routing path, any node that handles the routing of the EXIT packet must immediately forward this packet to the next hop. In addition, the routing node must purge its buffers of the packets associated with the EXIT packet's PKTID since they are no longer needed. Upon the reception of the EXIT packet, the sender can purge its buffers as well. Purging the buffers is done so as to release any resources tied up by the Data Transfer Session.

2.3 Error Recovery

If a fatal error occurs, the ABORT packet can be sent by the receiver or sender to indicate the failure of the transfer session. Every node on the routing path and the receiver of the ABORT packet must purge their buffers as well so as to free any resources that may have been allocated during the Data Transfer session. Any subsequent requests associated with a previously known and now unknown PKTID must

المنارة للاستشارات
www.manaraa.com

be responded to with an ABORT packet so as to restart the Data Transfer session if needed.

2.4 Timeout Handling

The INIT and CLOSE packets are essentially blocking commands since they are contingent on the receiver's timely response. This timeout period is based on the Estimated Total Delivery Time calculated in the Session Initiation Phase. Once a sender times out, they should resend the blocking command packet in an attempt to get a successful response. This may be repeated at most five times. If no response is received perhaps due to node failure along the transmission path, the sender may issue an ABORT command packet. In addition, the sender may store the packet for a later delivery attempt. Depending on the network topology, the sender may attempt to find another routing path to the destination. Timeouts are enforced in order to make sure that the sender does not wait indefinitely in case the receiver becomes unreachable.

المنارة للاستشارات

www.manaraa.com

3. Pipelined Transmission

We assume that the correct routing/relaying table is already established in the sensor network as stated earlier. Streaming data from a source node to the destination node along a predetermined path can be accomplished in either a coordinated or uncoordinated fashion. Here two transportation protocols are tested: one is a modified version of the Active Message (AM) protocol that is natively supported in TinyOS. AM is a random access protocol based on carrier sensing and can be regarded as a trimmed-down version of the IEEE802.11 MAC layer. This method allows nodes to compete for media access in order to transmit and is thus uncoordinated. This method offers a fair share of wireless channel access for wireless nodes, but is not optimized for any specific traffic pattern. Random media access is particularly inefficient for the pipelined transmission requirement. For example, if a relaying node repeatedly fails to acquire media access for transmission, which is very likely due to the nature of random access protocols, a pipeline bubble will be generated in its downstream nodes and cause significant delays.

The other transmission method is a Fix Scheduling algorithm (FS) which operates in a Time Division Multiplex (TDM) fashion. The transmission pattern is such that maximum parallel transmission is achieved along the path. A segment of the transmission schedule is shown in Figure 4. Details of the scheduling algorithm are discussed by Choi and Wang in [2]. We believe this approach is favorable for the data

www.manaraa.com

streaming application discussed here. The method allows efficient pipelined operations by maximizing media allocation along the data delivery path.

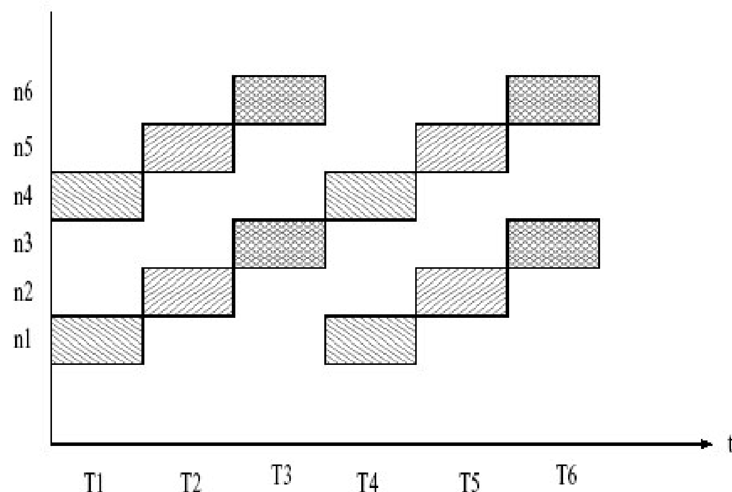


Fig. 4: FS scheduling pattern for 6 nodes.

3.1 Sensor nodes synchronization and TDM based Transmission

Using a TDM based transmission scheme requires that the sensors have synchronized clocks in order for them to transmit during their allocated transmission window. Typically wireless sensors do not have dedicated internal clocks for maintaining time. In order to accomplish synchronized transmission, we first have to solve the issue of keeping track of time in the nodes.

Synchronization procedure

المنارة للاستشارات
www.manaraa.com

To address this issue, we implemented a simple but effective technique. We use a Time Sync Broadcast packet to synchronize the nodes. Each node establishes a counter dedicated to keeping track of time in terms of its processor's clock ticks. When the nodes receive this packet, they reset their counters based on the information in the Time Sync Packet. It can be foreseen that due to propagation delay, some nodes may receive the packet before others, but analysis results show that this is negligible since the difference is in the micro second range and our TDM transmission scheme requires accuracy in the millisecond range.

Table I summarizes the results of the Time Synchronization experiment using 3 independent nodes. Initially each node determines a local time randomly. Five seconds after the synchronization procedure, the jitter at each node with respect to the "standard time" on the PC base station is zero.

المنارة للاستشارات
www.manaraa.com

	Node 1	Node 2	Node 3
initial	1000	400	450
5 secs	0	0	0
10 secs	1	2	2
30 secs	1	2	1
1 min	1	2	1

Table 1: Synchronized local time at sensor nodes

As time goes by, we observe that the jitter is slightly increased, but steadily maintained within 3 usec range. Since the scheduling protocol requires sub-millsecond level accuracy to support a maximum data rate of 76.8 kbps, the implemented local time service and the synchronization protocol meet the required precision for packet scheduling. For high data rate (such as MicaZ) transceivers, the timing service can still be used while maintaining good efficiency. However, the size of the packet must be increased accordingly.

It can be noted that with the passage of time, some of the sensor's clocks begin to get out of sync with the others. To get around this problem, we adopted a policy of resyncing the clocks every 1 sec so as to ensure that the nodes remain synchronized.

المنارة للاستشارات

www.manaraa.com

With the node clocks synchronized, each node is assigned a strict transmission slot. The time slices allocated should take into consideration the data transfer rate of the sensor in addition to the packet length it is transmitting. A simple formula that takes into consideration the issues listed above is shown below. It can be used to calculate the Minimum Time Slice Size (MinimumSliceSizems).

$$MinimumSliceSize_{ms} = ceiling((8000 \times packetsize_{bytes})/bitrate)$$

Once the time slots have been allocated, nodes can then transmit in a synchronized fashion according to their schedules with virtually zero probability of packet collision, thus producing a pipelined transmission sequence.

المنارة للاستشارات

www.manaraa.com

4. Retransmission Policy and Buffer Management

Packet drop handling is critical to the sustenance of high network throughput. In WSN, the problem is magnified severely due to the quick exhaustion of transmission buffers on the onset of such events. This also requires the transportation protocol to provide a recovery mechanism for packets lost due to node failure.

4.1 Retransmission Policy

The retransmission policy of dropped packets is closely tied to the buffer management scheme. The proposed application-aware retransmission policy is integrated into the Distributed Buffer Management Scheme in [17]. With a distributed buffer scheme, each relaying node might have several independent buffers for each connection. When retransmission is required after node failure, the upstream node needs to determine the order of packet retransmission so as to minimize the application-level delay. Based on the fragmentation information for each connection, the upstream node can estimate the pending transmission for each connection. We thus propose the following retransmission policy:

- Let P_i be the set of primary buffers and S_i be the set of secondary buffers for connection i. If P_i U S_i contain a complete fragmentation set, this connection should be selected for retransmission first.
- Otherwise, if none of the concurrent connections has a complete fragmentation set, the connection with the least pending fragment should be selected. Let X_i be

المنارة للاستشارات
www.manaraa.com

the minimum sequence number of P_i. The next retransmission will be awarded to the packet with sequence number $maxX_i$.

4.2 Buffer Management Policy

The distributed buffer management policy is fully discussed in [17] and is summarized here:

- For any node i, if the sequence number of the current head-of-buffer is X, it always buffers packet $\{(floor(X/M) - j).M + i\%M | j = 1 ...M\}$ in its secondary buffer.
- For node i, if a packet in the primary buffer satisfies the condition in policy 1, it will be moved to the secondary buffer right after it is successfully transmitted to the next hop.

Figure 5 illustrates the buffer distribution for one connection in a six-node path: packets are buffered along the routing nodes whose primary buffer is mirrored in the secondary buffers of the preceding routing nodes.

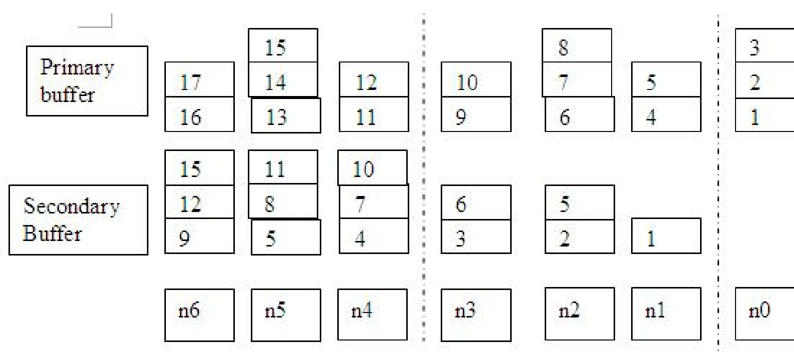


Figure 5: Distributed Buffer Overview [17]

www.manaraa.com

5. Hardware and Software Platform Details

The preliminary experiments are designed to evaluate the baseline performance. One wireless sensor node is designated as an information source (S node) where it captures still pictures periodically. The picture data is then relayed through a 5-hop path to a destination node (D node). For simplicity, the routing table is hard-coded in each node to allow efficient relaying. Some important node parameters are listed below:

- Device CPU Clock rate: 8 MHz
- Operating System: TinyOS
- Sensor Platform: Mica2
- Available buffer size: 32 Kbytes
- Buffer Size: 5 (Primary), 3 (Secondary)
- ISM Frequency: 900MHz

The preliminary experiments are conducted in a closed building environment. The transmission power of each node is tuned to a low level, about 0.5 mill-watts.

المنارة للاستشارات
www.manaraa.com

6. Experiment Setup

In order to evaluate the performance of our algorithm, we implemented the Light Weight Data Transportation Protocol using the Pipelined Transmission and Buffer Management scheme based on the methodology in [15]. We compared this to an implementation of stateless fragmentation with conventional buffer management techniques using CSMA/CD medium contention techniques.

6.1 Test Setup Instructions

In order to program the wireless sensors, we use AVR Studio. This is a programming platform provided by Atmel, the manufacturer of the microcontroller used in the wireless sensors. Figure 6 below shows a window of AVR Studio.

www.manaraa.com

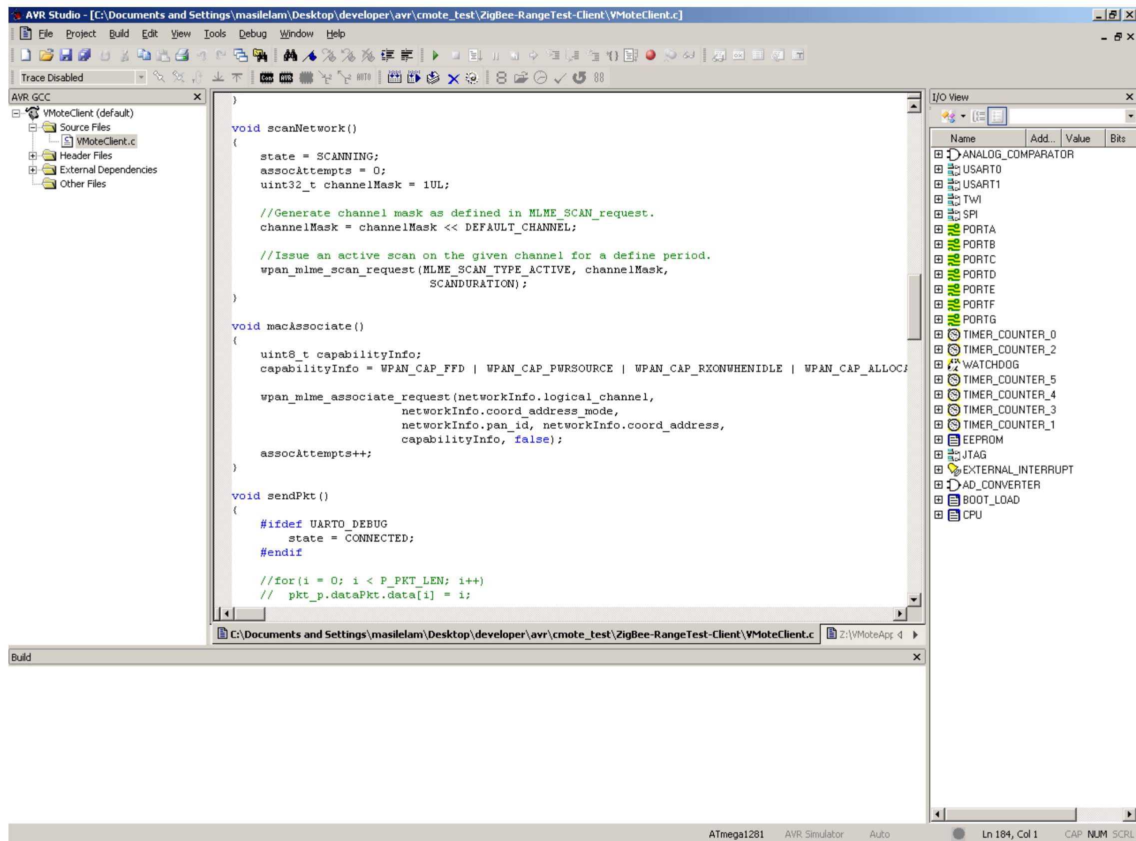


Figure 6: AVR Studio Window

Once the embedded program has been compiled, it can be uploaded on the wireless sensor using AVR Studio again. Figure 7 illustrates this procedure.

www.manaraa.com

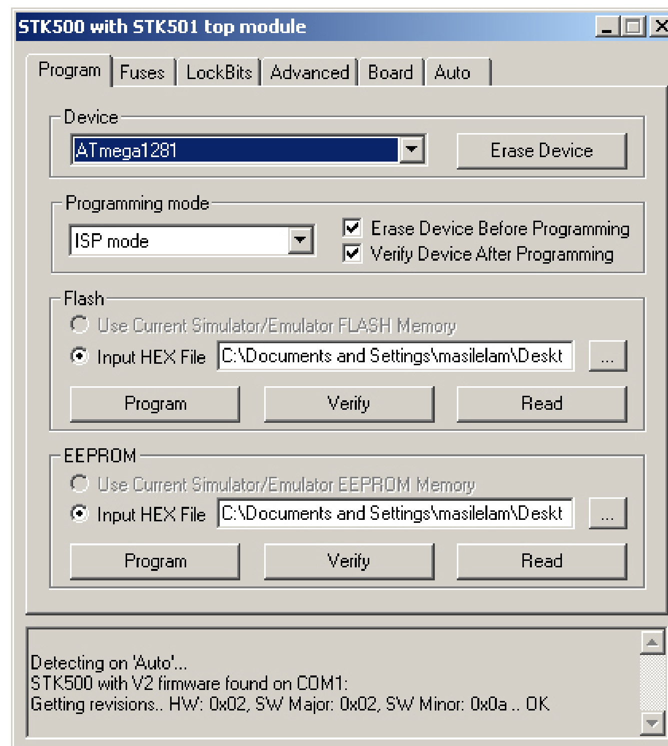


Figure 7: AVR Studio Device Programming Window

Once the program has been uploaded into the device, it will remain there until the device is reprogrammed again. The test files are located in the following location as listed below in Figure 8.

المنارة للاستشارات
www.manaraa.com

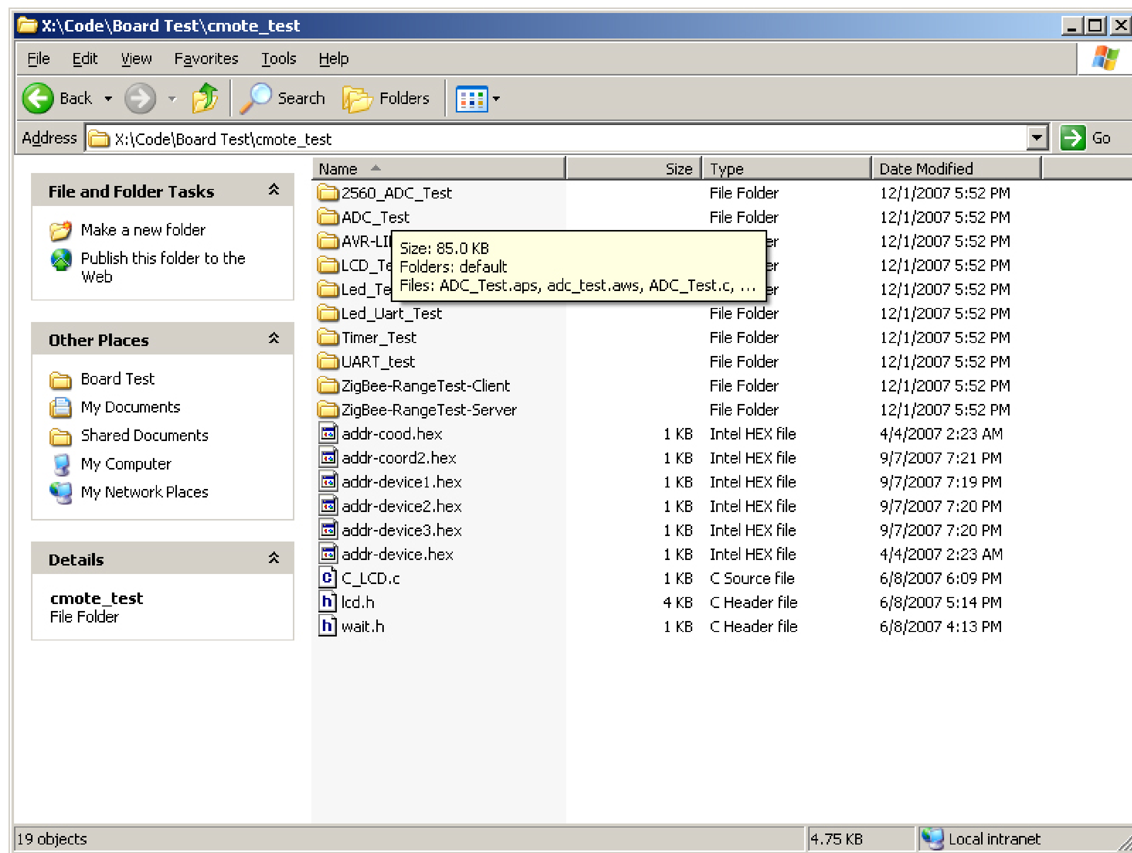


Figure 8: Test file location – X:\Code\Board Test\cmote_test

These test files were used to evaluate our sensor platform. All the sensor schematic files and Gerber PCB generation files are located in the repository as illustrated by Figure 9 and 10 below.

المنارة للاستشارات
www.manaraa.com

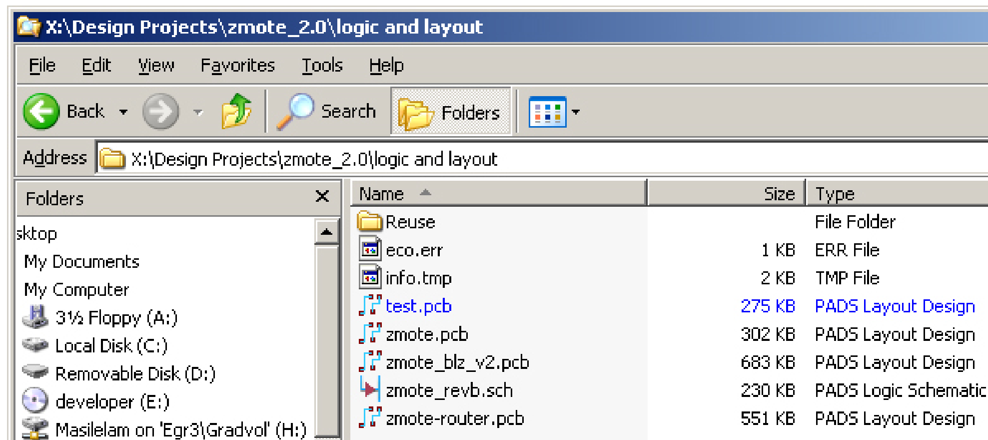
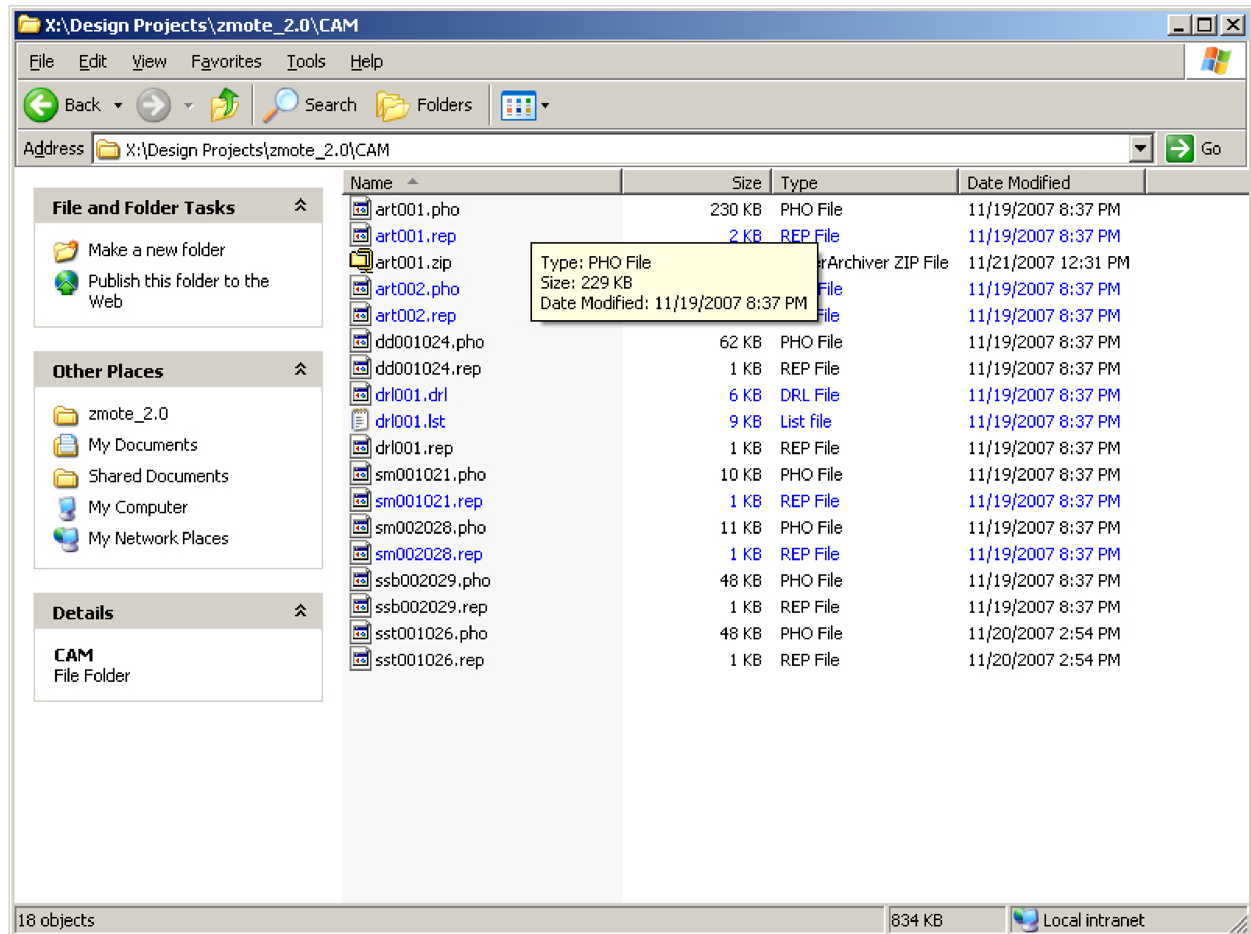


Figure 9: Schematic and PCB file location – X:\Design Projects\zmote_2.0\logic and layout



المنارة للاستشارات
www.manaraa.com

Figure 10: Gerber File location – X:\Design Projects\zmote_2.0\CAM

The final board that was created and tested is shown in Figure 1.b and its PCB Layout file is show in Figure 11 below. The software being used to view the board layout is PADS Layout.

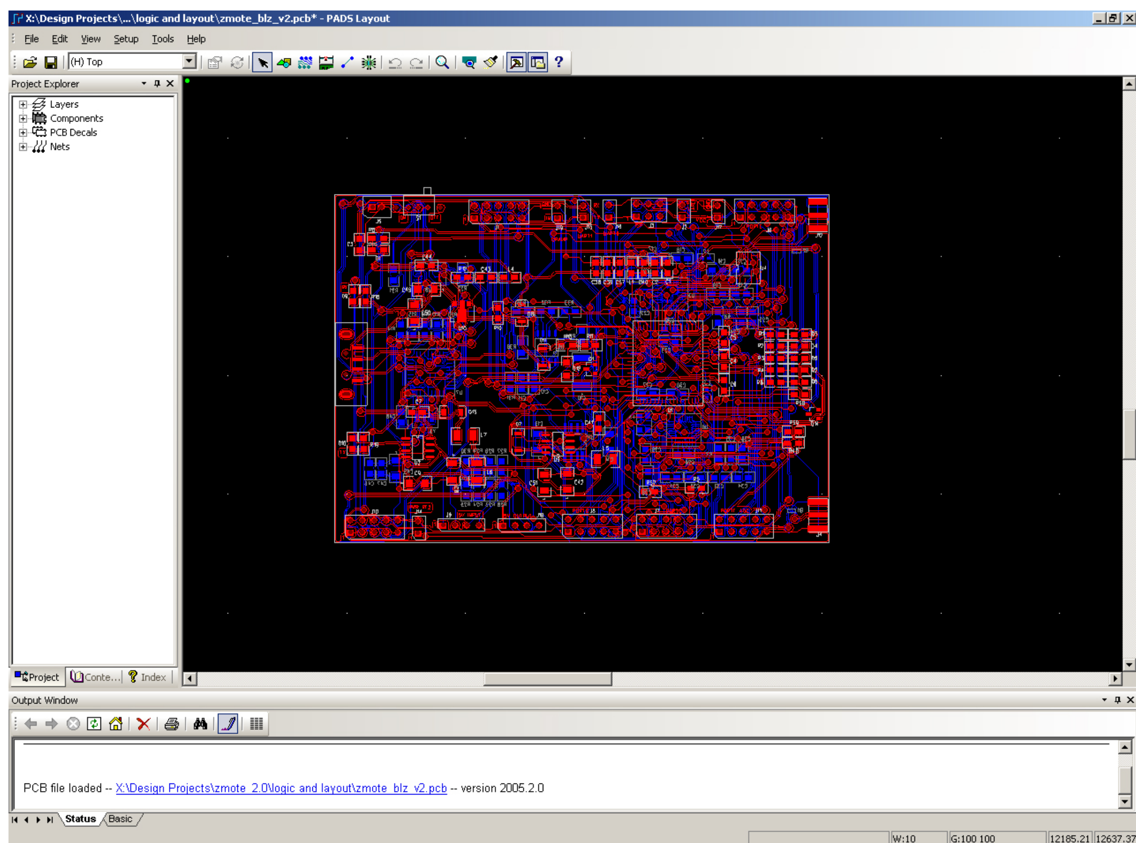


Figure 11: Sensor PCB Layout View

6.2 Implementation Details

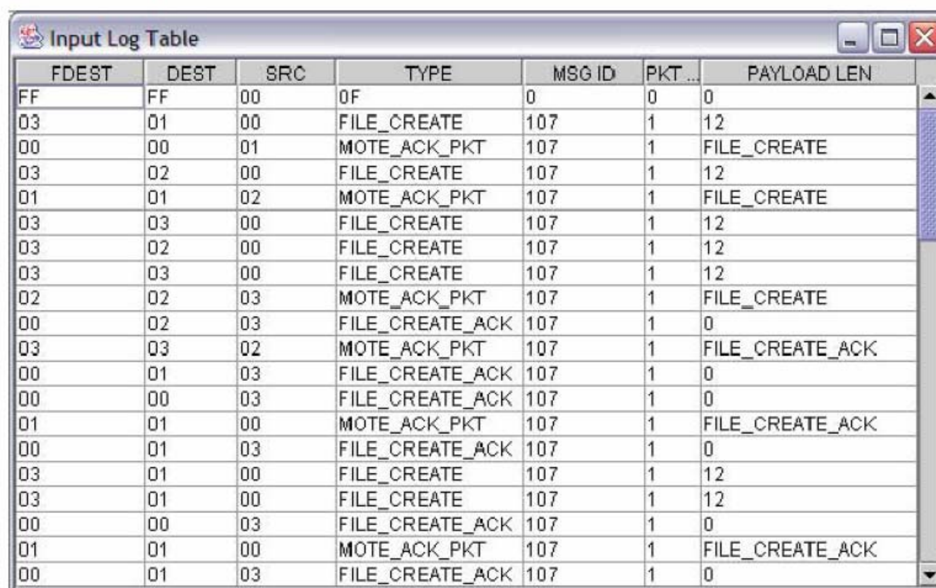
In our experiment setup, all the nodes in the network are time synchronized using the procedure that was defined in the Pipelined Transmission section. Five routing nodes are

المنارة للاستشارات
www.manaraa.com

used in addition to the source and sink. The source node contains a JPEG image fragmented by the PC attached to it. The JPEG image is fragmented into multiple 228-byte data payloads which are transmitted according to the schedule. The 7-node transmission path is tested using different data rates up to 200 Kbps in order to observe the performance of the algorithm.

Each sensor is programmed with the NesC code provided in the appendix. This code contains the implementation of the Lightweight Data Transportation Protocol and the buffer management techniques in addition to the time synchronization code. They are programmed with an individual address to uniquely identify each node. This can be achieved with the following command where 1 is the address of the node:

$ make install.1 mica2



Input Log Table

FDEST	DEST	SRC	TYPE	MSG ID	PKT ...	PAYLOAD LEN
FF	FF	00	0F	0	0	0
03	01	00	FILE_CREATE	107	1	12
00	00	01	MOTE_ACK_PKT	107	1	FILE_CREATE
03	02	00	FILE_CREATE	107	1	12
01	01	02	MOTE_ACK_PKT	107	1	FILE_CREATE
03	03	00	FILE_CREATE	107	1	12
03	02	00	FILE_CREATE	107	1	12
03	03	00	FILE_CREATE	107	1	12
02	02	03	MOTE_ACK_PKT	107	1	FILE_CREATE
00	02	03	FILE_CREATE_ACK	107	1	0
03	03	02	MOTE_ACK_PKT	107	1	FILE_CREATE_ACK
00	01	03	FILE_CREATE_ACK	107	1	0
00	00	03	FILE_CREATE_ACK	107	1	0
01	01	00	MOTE_ACK_PKT	107	1	FILE_CREATE_ACK
00	01	03	FILE_CREATE_ACK	107	1	0
03	01	00	FILE_CREATE	107	1	12
03	01	00	FILE_CREATE	107	1	12
00	00	03	FILE_CREATE_ACK	107	1	0
01	01	00	MOTE_ACK_PKT	107	1	FILE_CREATE_ACK
00	01	03	FILE_CREATE_ACK	107	1	0

المنارة للاستشارات
www.manaraa.com

Figure 12: Session Capture from the Snooper Node

Initially the source and sink are powered up. An additional "Snooper" node is powered up whose function is to record and monitor all traffic over the network. Figure 12 illustrates a sample session capture. The data captured by this program was used for the protocol evaluation. This node is strategically placed in order to eavesdrop on all traffic between nodes. This node is also attached to a PC in order to report all network activity in real time. A Java program attached to the source node is used to send a broadcast time sync message in order to begin the node synchronization. Once the nodes are time synchronized, the data transfer can now begin by running a Java program from the source node. This program fragments a packet and dispatches it for transportation. One of the images which was successfully transferred by the protocol is illustrated in Figure 13. This image is 4KB in size. The file is then monitored as it is transferred and all necessary statistics are recorded. This experiment is repeated various times and averages are collected for the statistics gathered. The analysis of this experiment is presented in the next section.

Figure 13: Sample Successfully Transported Image

المنارة للاستشارات
www.manaraa.com

6.3 Source Code Outline

The following source code files were used to perform the evaluation of our method. In addition, descriptions of each source code file are included. The files include Java and NesC files which are included in full in the appendix below.

FileTransferRouter.nc - This is the configuration NesC file responsible for setting up all the required software components for the Mica2 sensor.

FileTransferRouterM.nc – This file contains the main implementation of the protocol and buffer management system in addition to the time synchronization algorithm. This code is implemented on the Mic2 sensor.

BCastPkt.java – This program is used to broadcast packets to all sensors. This includes time sync packets and abort packets.

FileAgent.java – This interface enforces the Java protocol handlers to implement critical message reception routines.

FileMsg.java – This file containts the logic and rules that represent the packet. It also contains methods for extracting different portions of data from the packet.

FileTransferR.java – This is the driver program that controls all packets received at the destination node and is responsible for implementing all the receiver responses and requests dictated by the protocol.

FileTransferS.java – This is the driver program for the source node. It is responsible for all the logic for transferring the file and implements all the rules dictated by the protocol.

المنارة للاستشارات
www.manaraa.com

FTConstants.java – This file defines all the protocol constants.

FTProtocolHandler.java – This class enforces the driver programs to implement all critical session control information in addition to timeout and error handling.

FTRFile.java – This class stores the reconstructed fragmented file received.

FTSFile.java – This class is responsible for fragmenting the source file and storing it for future operations.

Int2Byte.java – This is a utility class for converting between integers and byte arrays.

LogWindow.java – This program simply logs all radio activity from all the sensor nodes.

PacketInjector.java – This program was used to inject corrupt or duplicate packets for testing and protocol evaluation purposes.

Serial2Mote.java – This class is the interface between the Mica2 sensors and the PCs. It processes all data between the two networks.

Snooper.java – This is the driver program run on the snooper PC for monitoring all network activity and data transfer. This program also helped collect all the necessary statistics for the protocol evaluation.

المنارة للاستشارات

www.manaraa.com

7. Protocol Evaluation

It is widely known that the packet drop ratio is closely related to the transmission data rate. The retransmission rate in our analysis is directly correlated to the packet drop ratio. Thus the proposed scheme is tested under different packet drop ratios by changing the physical layer data rate (time slice size). The observed data rate vs. the retransmission rate relationships are summarized in Table 1 and Table 2 for the Lightweight Data Transportation Protocol and the Stateless Fragmentation respectively.

Data Rate (Kbps)	Retransmission Rate(%)
200	50.45
125	13.65
110	9.8
100	0.1
50	0.1

Table 2: Lightweight Data Transportation Protocol Results: Data Rate vs. Retransmission Rate

المنارة للاستشارات
www.manaraa.com

Data Rate(Kbps)	Retransmission Rate(%)
200	67.2
125	47.43
100	23.41
50	12.16

Table 3: Stateless Fragmentation Results: Data Rate vs. Retransmission Rate

At 200 Kbps as expected, retransmission rates are high although our proposed scheme shows a significant 17.05% improvement over stateless fragmentation. The improvement at 100 Kbps significantly increases to about 23.4% efficiency over stateless fragmentation with virtually no packet loss. It is also observed that the use of stateless fragmentation performs poorly at a baseline data rate of 100 Kbps even though its performance can be improved by further reducing the data transmission rate, which may be undesirable. Even when reduced to 50Kbps, the performance of the stateless fragmentation is still very poor with almost 1 in every 10 packets being lost. It must be observed that our approach still produces impressive results even when the data rate has been lowered further.

المنارة للاستشارات
www.manaraa.com

8. Conclusion and Future Work

The Lightweight Data Transportation Protocol shows significant benefits for the transportation of large data units across sensor networks. It provides guaranteed data delivery with minimal retransmission overhead although its performance needs to be investigated over large networks. In addition, power considerations are yet to be discussed. The usage of efficient buffer management techniques increases the efficiency reducing the end-to-end retransmission cost and delay.

Pipelined transmission is a preferred method for large multimedia data transportation in wireless sensor networks. Our work shows that suitable flow control. In addition, our novel secondary buffer management effectively reduces the high retransmission overhead due to the unique nature of wireless sensor networks.

Time synchronization is a topic of study on its own. The method used in the experiments outlined above needs further improvement. Perhaps implementing the algorithm outlined in [16] would be a better approach to solving this problem.

المنارة للاستشارات
www.manaraa.com

List of References

[1] S. Patel, K. Lorincz, R. Hughes, N. Huggins, J. Growdon, M. Welsh and P Bonato, "Analysis of Feature Space for Monitoring Persons with Parkin son's Disease With Application to a Wireless Wearable Sensor System", in Proc. of the 29th IEEE EMBS Annual International Conference, Lyon, France, August 2007

[2] E. Hughes, M. Masilela, P. Eddings, A. Rafiq, C. Boanca and R. Merrell, "VMote: AWearableWirelessHealthMonitoringSystem",in Proc. 9th International Conference on e-Health Networking, Application and Services, Taipei, Taiwan, 2007.

[3] P. Bauer, M. Sichitiu, R. Istepanian and K. Premaratne, "The Mobile Patient: Wireless Distributed Sensor Networks for Patient Monitoring and Care", in Proc. of 3rd Conference on Information Technology Applications in Biomedicine, Arlington, VA, 2000

[4] I. E. Lamprinos, A. Prentza, E. Sakka and D. Koutsouris, "Energy-efficient MAC Protocol for Patient Personal Area Networks" in Proc. IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai,China, 2005.

[5] IEEE Standard 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-rate Wireless Personal Area Networks(LR-WPANs), 2003.

[6] X. Liang and I. Balasingham , "Performance Analysis of the IEEE 802.15.4 based ECG Monitoring Network", in Proc. 7th IASTED Internation Conference on Wireless and Optical Communications, Montreal, Quebec, Canada, 2007.

المنارة للاستشارات

www.manaraa.com

[7] Transmission Control Protocol, DARPA Internet Program Protocol Specification, Available: http://www.ietf.org/rfc/rfc793.txt.

[8] Xiaohua Luo, Kougen Zheng, Yunhe Pan and Zhaohui Wu, “ATCP/IP implementation for wireless sensor networks” , Proc. IEEE International Conference on Systems, Man and Cybernetics, Hague, Netherlands, 2004.

[9] O.B. Akan and I.F.Akyildiz, “Event-to-Sink Reliable Transport in Wireless Sensor Networks,” IEEE/ACM Transactions on Networking, 2005.

[10] F. Stann and J. Heidemann, “RMST: Reliable data transport in sensor networks,” in Proc. IEEE SNPA 03, Anchorage, Alaska, 2003.

[11] M. C. Vuran and I. F. Akyildiz, “Spatial Correlation-based Collaborative Medium Access Control in Wireless Sensor Networks,” IEEE/ACM Transactions on Networking, 2006.

[12] W.Ye, J.Heidemann, and D.Estrin, “An energy-efficient MAC protocol for wireless sensor networks”, in Proc. 21st Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, 2002.

[13] C.Intanagonwiwat, R.Govindan, D.Estrin, J.Heidemann and F.Silva, “Directed diffusion for wireless sensor networking,” IEEE/ACM Transactions on Networking, 2003.

[14] K. Seada, M. Zuniga, A. Helmy and B. Krishnamachari, “Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks,” in in Proc. ACM Sensys 04, Baltimore, Maryland, 2004.

المنارة للاستشارات
www.manaraa.com

[15] J.Wang, M.Masilela and J.Liu, "Distributed Retransmission Buffer Recovery for Node Failure in Sensor Networks",in Proc. 7th IASTED International Conference on Wireless and Optical Communications, Montreal, Quebec, Canada, 2007.

[16] J.Greunen and J.Rabaey, "Lightweight Time Synchronization for Sensor Networks", in Proc. 2nd ACM International Conference on Wireless Sensor Networks and Applications, San Diego, CA, USA, 2003

المنارة للاستشارات

www.manaraa.com

Appendix A – Source Code

```
FileTransferRouter.nc
/**
 * @author Mbonisi Masilela
 *
 * Based on Snooper
 */


includes config;
includes DataMsg;

configuration FileTransferRouter
{
	provides interface ClockTick;
}

implementation
{
	components Main, FileTransferRouterM, PhyRadio, UART,
LedsC, ClockC;/*, HPLCC1000M, CC1000ControlM;*/

	ClockTick = FileTransferRouterM;
	Main.StdControl -> FileTransferRouterM;
	FileTransferRouterM.PhyControl -> PhyRadio;
	FileTransferRouterM.PhyComm -> PhyRadio;
	FileTransferRouterM.PhyStreamByte -> PhyRadio;
	FileTransferRouterM.UARTControl -> UART;
	FileTransferRouterM.UARTComm -> UART;
	FileTransferRouterM.Leds -> LedsC;
	FileTransferRouterM.Clock -> ClockC;
	//FileTransferRouterM.CarrierSense -> PhyRadio;
	//FileTransferRouterM.ClockSlice -> ClockC;

	//FileTransferRouterM.CC1000Control -> CC1000ControlM;

	//FileTransferRouterM.CC1000StdControl ->
CC1000ControlM;
	//CC1000ControlM.HPLChipcon -> HPLCC1000M.HPLCC1000;
}
```

www.manaraa.com

```
/**
   @author Mbonisi Masilela

   Preset 0: 12364bps
   This means that:
     i.        with a packet of 236, effective rate is
approximately 6 packets per sec
     ii.       it takes about 0.65 millisecs to transit a
byte of information
     iii. to transmit a packet of 236 bytes, a slice of at
least 154 ms would be recommended
*/

module FileTransferRouterM
{
     provides interface StdControl;
     provides interface ClockTick;

     uses
     {
          interface StdControl as PhyControl;
          interface PhyComm;
          interface PhyStreamByte;
          interface StdControl as UARTControl;
          interface ByteComm as UARTComm;
          interface Leds;
          interface Clock;
     }
}

implementation
{
     #ifndef TOS_UART_ADDR
          #define TOS_UART_ADDR 0x7E
     #endif

     #ifndef MOTE_BCAST_ADDR
          #define MOTE_BCAST_ADDR 0xFF
     #endif

     #ifndef BASE_NODE_ADDR
          #define BASE_NODE_ADDR 0x00
     #endif
```

www.manaraa.com

```
#define TIME_SLICE_INTERVAL 200

#define TIME_PACKETS_PER_SEC 5
#define RX_BUF_SIZE 4
#define TX_BUF_SIZE 5
#define BASE_SYNC_WAIT 1
#define ACK_TIMEOUT TIME_SLICE_INTERVAL * 9/10

//typedef unsigned long long int uint64_t;

AppPkt dataPkt;
uint8_t baseNodePendingReset;
uint8_t bstartTime;
uint8_t radioPacketIndex, uartPacketIndex,
uartSendIndex;
uint8_t syncFlag;
uint8_t checkNext;
uint8_t radioPacket[PACKET_SIZE];
uint8_t uartPacket[PACKET_SIZE];
uint8_t tempUartPacket[PACKET_SIZE];
uint32_t time;
uint32_t transmissionTime;

uint8_t MOTE_TIME_SLICE;
uint8_t CURRENT_TIME_SLICE;
uint8_t TRANSMIT_MODE;

uint8_t TIME_SYNCED;
uint8_t PENDING_ACK;
uint8_t CHANNEL_STATUS;
uint8_t PROCESSING_RADIO_DATA;
uint8_t PC;

uint8_t currentAckTicks;

uint16_t interval;
uint8_t bufIndex;

uint8_t nextTxIndex;
uint8_t nextEmptyIndex;
uint8_t pendingTxPkts;

AppPkt rxBuf[RX_BUF_SIZE];
AppPkt txBuf[TX_BUF_SIZE];
```

www.manaraa.com

```
task void sendPktRadio();
task void routePacket();

enum
{
	SYNC_DONE = 0,
	SYNC_NOT_DONE
};

enum
{
	TRUE = 0,
	FALSE,
	PROCESSING
};

enum
{
	CLEAR,
	BUSY
};

enum
{
	TOTAL_PACKET_LEN_INDEX = 0,
	SEQ_NUM_INDEX = 1,
	DEST_ADDR_INDEX = 2,
	SRC_ADDR_INDEX,
	PAYLOAD_LEN_INDEX,
	MSG_TYPE_INDEX,
	PACKET_ID_INDEX,
	PACKET_NUM_INDEX = 7,
	PACKET_NUM2_INDEX = 8,
	DATA_START_INDEX = 9,
	CRC_INDEX = (PACKET_SIZE - 2)
};

enum
{
	FILE_CREATE = 1,
	FILE_CREATE_ACK = 10,
	FILE_DATA = 2,
	FILE_CLOSE = 3,
```

www.manaraa.com

```
		FILE_CLOSE_ACK = 11,
		FILE_ERROR = 4,
		FILE_INFO_ERROR = 13,
		FILE_REREQUEST = 5,
		FILE_EXIT = 9,
		FILE_EXIT_ACK = 12,
		MOTE_SYNC_TIME = 14,
		MOTE_RESET_TIME = 15,
		MOTE_SEND_STATS = 16,
		MOTE_STATUS_DATA = 17,
		MOTE_ACK_PKT = 18
	};

	void add2RxBuf()
	{
		memcpy(&rxBuf[bufIndex], uartPacket,
sizeof(AppPkt));
		bufIndex++;

		if(bufIndex >= RX_BUF_SIZE)
			bufIndex =0;
	}

	void add2TxBuf(void* pkt)
	{
		/*if(pendingTxPkts < TX_BUF_SIZE)
		{
			memcpy(&txBuf[nextEmptyIndex], pkt,
sizeof(AppPkt));
			nextEmptyIndex++;

			if(nextEmptyIndex >= TX_BUF_SIZE)
				nextEmptyIndex = 0;

			pendingTxPkts++;
		}*/

		memcpy(&txBuf[0], pkt, sizeof(AppPkt));
		pendingTxPkts = 1;
	}

	void loadTxPkt()
	{
```

المنارة للاستشارات

www.manaraa.com

```
			//memcpy(&dataPkt, &txBuf[nextTxIndex], 
sizeof(AppPkt));
			memcpy(&dataPkt, &txBuf[0], sizeof(AppPkt));
			PENDING_ACK = TRUE;

			//if(dataPkt.data[MSG_TYPE_INDEX - 2] == 
MOTE_ACK_PKT)
			//	PENDING_ACK = FALSE;
		}

		void prepareForNextTxPkt()
		{
			PENDING_ACK = FALSE;
			pendingTxPkts = 0;
			/*pendingTxPkts--;
			if(pendingTxPkts > 0)
			{
				nextTxIndex++;
				if(nextTxIndex >= TX_BUF_SIZE)
					nextTxIndex = 0;

				//if(pendingTxPkts > 0)
					//pendingTxPkts--;
			}*/
		}

		int8_t findPkt(uint8_t pid, uint8_t pnum1, uint8_t 
pnum2)
		{
			int8_t i = 0;
			while(i < RX_BUF_SIZE)
			{
				if(rxBuf[i].data[PACKET_NUM_INDEX - 2] == 
pnum1
					&& rxBuf[i].data[PACKET_NUM2_INDEX - 2] 
== pnum2
					&& rxBuf[i].data[PACKET_ID_INDEX - 2] 
== pid
					&& rxBuf[i].data[MSG_TYPE_INDEX - 2] == 
FILE_DATA)
					return i;
				i++;
			}
			return -1;
```

www.manaraa.com

```
	}

	void sendRPkt(uint8_t dadd, uint8_t msgid, uint8_t
pktNum1, uint8_t pktNum2)
	{
		dataPkt.phyhdr.hdr.length = sizeof(AppPkt);

		dataPkt.phyhdr.seqNo = dadd;
		dataPkt.data[DEST_ADDR_INDEX - 2] = dadd;
		dataPkt.data[SRC_ADDR_INDEX - 2] =
TOS_LOCAL_ADDRESS;
		dataPkt.data[MSG_TYPE_INDEX - 2] =
FILE_REREQUEST;
		dataPkt.data[PACKET_NUM_INDEX - 2] = pktNum1;
		dataPkt.data[PACKET_NUM2_INDEX - 2] = pktNum2;
		dataPkt.data[PACKET_ID_INDEX - 2] = msgid;
		dataPkt.data[PAYLOAD_LEN_INDEX - 2] = PC;

		add2TxBuf(&dataPkt);
	}

	void sendAck(uint8_t src, uint8_t pid, uint8_t pn1,
uint8_t pn2, uint8_t msgT)
	{
		dataPkt.phyhdr.hdr.length = sizeof(AppPkt);

		dataPkt.phyhdr.seqNo = src;
		dataPkt.data[DEST_ADDR_INDEX - 2] = src;
		dataPkt.data[SRC_ADDR_INDEX - 2] =
TOS_LOCAL_ADDRESS;
		dataPkt.data[MSG_TYPE_INDEX - 2] = MOTE_ACK_PKT;
		dataPkt.data[PACKET_ID_INDEX - 2] = pid;
		dataPkt.data[PACKET_NUM_INDEX - 2] = pn1;
		dataPkt.data[PACKET_NUM2_INDEX - 2] = pn2;
		dataPkt.data[PAYLOAD_LEN_INDEX - 2] = msgT;

		//add2TxBuf(&dataPkt);

		//Block until ack is sent
		while(call PhyComm.txPkt(&dataPkt,
sizeof(AppPkt)) != SUCCESS);
	}

	void resetV()
```

المنارة للاستشارات

www.manaraa.com

```
    {
          interval = 1000/TIME_PACKETS_PER_SEC;

          radioPacketIndex = 0;
          uartPacketIndex = 0;
          uartSendIndex = 0;
          syncFlag = SYNC_NOT_DONE;
          checkNext = FALSE;
          time = 0;
          currentAckTicks = 0;
          transmissionTime = 0;
          bufIndex = 0;
          nextTxIndex = 0;
          nextEmptyIndex = 0;
          pendingTxPkts = 0;

          CURRENT_TIME_SLICE = 0;
          TRANSMIT_MODE = FALSE;
          TIME_SYNCED = FALSE;
          PENDING_ACK = FALSE;
          PROCESSING_RADIO_DATA = FALSE;
          CHANNEL_STATUS = BUSY;
          PC = 0;
          baseNodePendingReset = FALSE;
          bstartTime = 0;
    }

    void syncTime()
    {
          atomic
          {
                resetV();

                time = 0;
                CURRENT_TIME_SLICE = 0;
                call Clock.setRate(TOS_I1000PS,
TOS_S1000PS);
                call Leds.redOff();

                TIME_SYNCED = TRUE;
                PC = 0;
                baseNodePendingReset = FALSE;
          }
    }
```

www.manaraa.com

```
	task void sendPktRadio()
	{
		PROCESSING_RADIO_DATA = TRUE;

		if(TRANSMIT_MODE == TRUE && PENDING_ACK == FALSE
&& TIME_SYNCED == TRUE)
		{
			loadTxPkt();
			if(call PhyComm.txPkt(&dataPkt,
sizeof(AppPkt)) == SUCCESS)
			{
				call Leds.greenToggle();
			}
			else
				call Leds.redToggle();
		}
		PROCESSING_RADIO_DATA = FALSE;
	}


	task void routePacket()
	{
		int8_t idx = 0;

		if(uartPacket[DEST_ADDR_INDEX] ==
TOS_LOCAL_ADDRESS
			&& uartPacket[MSG_TYPE_INDEX] ==
MOTE_ACK_PKT
			&& uartPacket[SEQ_NUM_INDEX] ==
TOS_LOCAL_ADDRESS)
		{
			if(uartPacket[PACKET_ID_INDEX] ==
txBuf[nextTxIndex].data[PACKET_ID_INDEX - 2] &&
			   uartPacket[PACKET_NUM_INDEX] ==
txBuf[nextTxIndex].data[PACKET_NUM_INDEX - 2] &&
			   uartPacket[PACKET_NUM2_INDEX] ==
txBuf[nextTxIndex].data[PACKET_NUM2_INDEX - 2] &&
			   uartPacket[PAYLOAD_LEN_INDEX] ==
txBuf[nextTxIndex].data[MSG_TYPE_INDEX - 2] &&
			   uartPacket[SRC_ADDR_INDEX] ==
txBuf[nextTxIndex].data[DEST_ADDR_INDEX - 2])
			   {
```

www.manaraa.com

```
                         prepareForNextTxPkt();

                         call
UARTComm.txByte(TOS_UART_ADDR);
                         uartSendIndex = 0;

                  }
          }
          else if(uartPacket[DEST_ADDR_INDEX] ==
TOS_LOCAL_ADDRESS)
          {
               if(uartPacket[SRC_ADDR_INDEX] <
TOS_LOCAL_ADDRESS)
                    idx = TOS_LOCAL_ADDRESS - 1;
               else
                    idx = TOS_LOCAL_ADDRESS + 1;
               sendAck(idx,
                         uartPacket[PACKET_ID_INDEX],
                         uartPacket[PACKET_NUM_INDEX],
                         uartPacket[PACKET_NUM2_INDEX],
                         uartPacket[MSG_TYPE_INDEX]);
               idx = 0;
               if(uartPacket[MSG_TYPE_INDEX] ==
FILE_REREQUEST && uartPacket[SEQ_NUM_INDEX] ==
TOS_LOCAL_ADDRESS)
               {
                    if(TOS_LOCAL_ADDRESS == BASE_NODE_ADDR)
                    {
                         call
UARTComm.txByte(TOS_UART_ADDR);
                         uartSendIndex = 0;

                    }
                    else
                    {
                         idx =
findPkt(uartPacket[PACKET_ID_INDEX],
uartPacket[PACKET_NUM_INDEX],
uartPacket[PACKET_NUM2_INDEX]);
                         if (idx == -1)

     sendRPkt(uartPacket[DEST_ADDR_INDEX]-1,

uartPacket[PACKET_ID_INDEX],
```

المنارة للاستشارات
www.manaraa.com

```
uartPacket[PACKET_NUM_INDEX],

uartPacket[PACKET_NUM2_INDEX]);
                         else
                         {
                              add2TxBuf(&rxBuf[idx]);
                         }
                    }
               }
               else if(uartPacket[SEQ_NUM_INDEX] ==
TOS_LOCAL_ADDRESS)
               {
                    call UARTComm.txByte(TOS_UART_ADDR);
                    uartSendIndex = 0;
               }
               else
               {
                    if(uartPacket[SEQ_NUM_INDEX] >
TOS_LOCAL_ADDRESS)
                         uartPacket[DEST_ADDR_INDEX]++;
                    else
                         uartPacket[DEST_ADDR_INDEX]--;

                    if(uartPacket[MSG_TYPE_INDEX] ==
FILE_DATA)
                         add2RxBuf();

                    add2TxBuf(uartPacket);
               }
          }
          else if(uartPacket[DEST_ADDR_INDEX] ==
MOTE_BCAST_ADDR)
          {
               if(uartPacket[MSG_TYPE_INDEX] ==
MOTE_SYNC_TIME ||
                  uartPacket[MSG_TYPE_INDEX] ==
MOTE_RESET_TIME)
               {
                    syncTime();
               }
          }
     }
```

المنارة للاستشارات
www.manaraa.com

```
command result_t StdControl.init()
{
     //uint8_t i;

     call UARTControl.init();
     call PhyControl.init();
     call Leds.init();

     resetV();

     return SUCCESS;
}

command result_t StdControl.start()
{
     call UARTControl.start();
     call PhyControl.start();
     pendingTxPkts = 0;
     MOTE_TIME_SLICE = TOS_LOCAL_ADDRESS % 3;
     call Clock.setRate(TOS_I1000PS, TOS_S1000PS);

     return SUCCESS;
}

async event result_t Clock.fire()
{
     time++;

     //startCarrierSense();

     if(time % TIME_SLICE_INTERVAL == 0)
          CURRENT_TIME_SLICE++;

     if(CURRENT_TIME_SLICE == 3)
          CURRENT_TIME_SLICE = 0;

     if(CURRENT_TIME_SLICE == MOTE_TIME_SLICE)
     {
          TRANSMIT_MODE = TRUE;

          call Leds.yellowOn();
          call Leds.redOff();

          if(PENDING_ACK == TRUE)
```

المنارة للاستشارات

www.manaraa.com

```
                    currentAckTicks++;
               else
                    currentAckTicks = 0;

               if(currentAckTicks > ACK_TIMEOUT &&
PENDING_ACK == TRUE)
               {
                    PENDING_ACK = FALSE;
                    currentAckTicks = 0;
               }

               if(time % 2 == 0)
                    signal ClockTick.clockTicked();

          }
          else
          {
               TRANSMIT_MODE = FALSE;
               call Leds.yellowOff();

               if(TIME_SYNCED == FALSE)
                    call Leds.redOn();
          }

          //For base station
     if(baseNodePendingReset == TRUE && BASE_SYNC_WAIT <=
(time - bstartTime) && TOS_LOCAL_ADDRESS == BASE_NODE_ADDR)
          syncTime();

     return SUCCESS;
    }

     default async event result_t ClockTick.clockTicked()
     {
          //call Leds.redOn();
     //if(PROCESSING_RADIO_DATA == TRUE && TRANSMIT_MODE ==
TRUE)
     if(TRANSMIT_MODE == TRUE && pendingTxPkts > 0 &&
PROCESSING_RADIO_DATA == FALSE && TIME_SYNCED == TRUE)
          post sendPktRadio();

          return SUCCESS;
     }
```

www.manaraa.com

```
	command result_t StdControl.stop()
	{
		call UARTControl.stop();
		call PhyControl.stop();
		return SUCCESS;
	}

	event result_t PhyComm.startSymDetected(void* pkt)
	{
		uartPacketIndex = 0;
		return SUCCESS;
	}

	event result_t PhyStreamByte.rxByteDone(char data)
	{
		//call Leds.yellowToggle();
		tempUartPacket[uartPacketIndex] = data;
		uartPacketIndex++;

		return SUCCESS;
	}

	event void* PhyComm.rxPktDone(void* packet, char
error)
	{
		memcpy(uartPacket, tempUartPacket,
sizeof(uartPacket));
		//memcpy(tempUartPacket, blank, PACKET_SIZE);
		uartPacketIndex = 0;

		post routePacket();

		return packet;
	}

	event result_t PhyComm.txPktDone(void* packet)
	{
		return SUCCESS;
	}

	async event result_t UARTComm.txByteReady(bool
success)
	{
		if(uartSendIndex < PACKET_SIZE)
```

المنارة للاستشارات
www.manaraa.com

```
		{
			call 
UARTComm.txByte(uartPacket[uartSendIndex]);
			uartSendIndex++;
		}
		else
		{
			//memcpy(uartPacket, blank, PACKET_SIZE);
			uartSendIndex = 0;
		}

		return SUCCESS;
	}

	async event result_t UARTComm.rxByteReady(uint8_t 
data, bool error, uint16_t strength)
	{
		if(TOS_LOCAL_ADDRESS == BASE_NODE_ADDR)
			call Leds.yellowToggle();

		if(syncFlag == SYNC_NOT_DONE)
		{
			if(checkNext == FALSE)
			{
				if(data == TOS_UART_ADDR)
				{
					checkNext = TRUE;
				}
			}
			else
			{
				if(data == sizeof(AppPkt))
				{
					radioPacketIndex = 0;
					syncFlag = SYNC_DONE;
					//memcpy(radioPacket, blank, 
PACKET_SIZE);

					radioPacket[0] = sizeof(AppPkt);
					radioPacketIndex++;
				}
				else if(data == TOS_UART_ADDR)
				{
					checkNext = TRUE;
				}
```

المنارة للاستشارات
www.manaraa.com

```
                    else
                    {
                         checkNext = FALSE;
                         syncFlag = SYNC_NOT_DONE;
                         //call Leds.redToggle();
                    }
               }
          }
          else
          {
               radioPacket[radioPacketIndex] = data;

               if(radioPacketIndex == (PACKET_SIZE - 1))
               {
                    radioPacket[SEQ_NUM_INDEX] =
radioPacket[DEST_ADDR_INDEX];

                    if(radioPacket[DEST_ADDR_INDEX] !=
MOTE_BCAST_ADDR)
                    {
                         if(radioPacket[DEST_ADDR_INDEX] >
TOS_LOCAL_ADDRESS)
                              radioPacket[DEST_ADDR_INDEX]
= TOS_LOCAL_ADDRESS + 1;
                         else
                              radioPacket[DEST_ADDR_INDEX]
= TOS_LOCAL_ADDRESS - 1;

                         if(radioPacket[MSG_TYPE_INDEX] ==
FILE_REREQUEST)
                              radioPacket[SEQ_NUM_INDEX] =
radioPacket[DEST_ADDR_INDEX];

                         add2TxBuf(radioPacket);
                    }
                    else
                    {
                         if(radioPacket[MSG_TYPE_INDEX] ==
MOTE_SYNC_TIME ||
                            radioPacket[MSG_TYPE_INDEX] ==
MOTE_RESET_TIME)
                         {
                            baseNodePendingReset = TRUE;
                            bstartTime = time;
```

المنارة للاستشارات
www.manaraa.com

```
                         }

                         call PhyComm.txPkt(radioPacket,
sizeof(AppPkt));
                    }

                    radioPacketIndex = 0;
                    syncFlag = SYNC_NOT_DONE;
                    checkNext = FALSE;
               }
               else
                    radioPacketIndex++;
          }
          return SUCCESS;
     }

     async event result_t UARTComm.txDone()
     {
          return SUCCESS;
     }
}
// end of implementation
```

المنارة للاستشارات
www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: BitUtils.java
  */
public class BitUtils
{
    public static int byteArrayToInt(byte[] b)
    {
          int offset = 0;
        int value = 0;
        for (int i = 0; i < b.length; i++)
        {
            int shift = (b.length - 1 - i) * 8;
            value += (b[i + offset] & 0x000000FF) << shift;
        }
        return value;
     }

     public static byte[] intToByteArray(int val)
     {
          byte[] mybyte = new byte[4];

          mybyte[0] =(byte)( val >> 24 );
          mybyte[1] =(byte)( (val << 8) >> 24 );
          mybyte[2] =(byte)( (val << 16) >> 24 );
          mybyte[3] =(byte)( (val << 24) >> 24 );

          return mybyte;
     }

     public static byte inttobyte(int val)
     {
          return (byte)((val << 24) >> 24 );
     }

     public static int bytetoint(byte b)
     {
          return (int)(b & 0xFF);
     }

     public static String byteArrayToHexString(byte in[])
     {
          String str = "";
```

المنارة للاستشارات

www.manaraa.com

```
            if (in == null || in.length <= 0)
                 return null;

            for(int i = 0; i < in.length; i++)
                 str += byteToHexString(in[i]);

            return str;
      }

      public static String byteToHexString(byte in)
      {
            byte ch = 0x00;
            int i = 0;

            String[] pseudo = {"0", "1", "2","3", "4", "5",
"6", "7",
                                       "8", "9", "A", "B", "C",
"D", "E","F"};

            StringBuffer out = new StringBuffer(2);

            ch = (byte) (in & 0xF0); // Strip off high nibble
            ch = (byte) (ch >>> 4);
            // shift the bits down
            ch = (byte) (ch & 0x0F);
            // must do this is high order bit is on!
            out.append(pseudo[ (int) ch]); // convert the
nibble to a String Character
            ch = (byte) (in & 0x0F); // Strip off low nibble
            out.append(pseudo[ (int) ch]); // convert the
nibble to a String Character
            out.append(" ");

            String rslt = new String(out);
            return rslt;
      }
}


/**
  * @author: Mbonisi Masilela
  * File: FileAgent.java
  */
```

www.manaraa.com

```
public interface FileAgent
{
     public void messageReceived(FileMsg msg);
}
```

المنارة للاستشارات

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: FileMsg.java
  */

/*
Bytes
     0 - DEST ADDR
     1 - SRC ADDR
     2 - PAYLOAD LEN
     3 - MSG TYPE
     4 - Packet ID
     5 - Packet Number
     6 - 230 Data (225 data byte payload)
*/

public class FileMsg
{
     byte[] msg;
     int dataStartIndex = 6;

     public FileMsg()
     {
          msg = new byte[FTConstants.PACKET_SIZE];
          msg[0] =
BitUtils.inttobyte(FTConstants.PACKET_SIZE);
     }

     public FileMsg(byte[] data) throws Exception
     {
          msg = new byte[FTConstants.PACKET_SIZE];

          if(data.length != FTConstants.PACKET_SIZE)
               throw new Exception("FileMsg ::: Your data
must match the standard packet size");
          else
               for(int i = 0; i < FTConstants.PACKET_SIZE;
i++)
                    msg[i] = data[i];

          if(msg[0] !=
BitUtils.inttobyte(FTConstants.PACKET_SIZE))
               throw new Exception("FileMsg ::: Invalid
Packet ... does not begin with the standard length");
```

www.manaraa.com

```
	}

	public void setDestAddr(byte addr)
	{
		msg[FTConstants.DEST_ADDR_INDEX] = addr;
	}

	public byte getDestAddr()
	{
		return msg[FTConstants.DEST_ADDR_INDEX];
	}

	public void setSrcAddr(byte addr)
	{
		msg[FTConstants.SRC_ADDR_INDEX] = addr;
	}

	public byte getSrcAddr()
	{
		return msg[FTConstants.SRC_ADDR_INDEX];
	}

	public void setLength(int len)
	{
		if(len > FTConstants.DATA_MSG_SIZE)
			len = FTConstants.DATA_MSG_SIZE;

		msg[FTConstants.PAYLOAD_LEN_INDEX] = 
BitUtils.inttobyte(len);//Byte.parseByte("" + len);
	}

	public int getLength()
	{
		return 
BitUtils.bytetoint(msg[FTConstants.PAYLOAD_LEN_INDEX]);
	}

	public void setMsgType(byte type)
	{
		msg[FTConstants.MSG_TYPE_INDEX] = type;
	}

	public byte getMsgType()
	{
```

www.manaraa.com

```
            return msg[FTConstants.MSG_TYPE_INDEX];
        }

        public void setMsgID(int id)
        {
            msg[FTConstants.PACKET_ID_INDEX] =
BitUtils.inttobyte(id);//Byte.parseByte("" + id);
        }

        public int getMsgID()
        {
            return
BitUtils.bytetoint(msg[FTConstants.PACKET_ID_INDEX]);
        }

        public void setPacketNumber(int num)
        {
            byte[] ibytes = BitUtils.intToByteArray(num);

            msg[FTConstants.PACKET_NUM_INDEX] =
ibytes[ibytes.length - 2];
            msg[FTConstants.PACKET_NUM_INDEX + 1] =
ibytes[ibytes.length - 1];
        }

        public int getPacketNumber()
        {
            byte[] ibyte = new byte[2];
            ibyte[0] = msg[FTConstants.PACKET_NUM_INDEX];
            ibyte[1] = msg[FTConstants.PACKET_NUM_INDEX + 1];

            return BitUtils.byteArrayToInt(ibyte);
        }

        public void setData(byte[] data)
        {
            for(int i = 0; i < data.length; i++)
                msg[FTConstants.DATA_START_INDEX + i] =
data[i];
        }

        public byte[] getData()
        {
            int len = getLength();
```

المنارة للاستشارات
www.manaraa.com

```
            int newdataindex = 0;
            byte[] newData = new byte[len];

            for(int i = FTConstants.DATA_START_INDEX;
newdataindex < len; i++, newdataindex++)
                  newData[newdataindex] = msg[i];

            return newData;
      }

      public byte[] getCRC()
      {
            byte[] crc = new byte[2];
            crc[0] = msg[FTConstants.CRC_INDEX + 1];
            crc[1] = msg[FTConstants.CRC_INDEX];

            return crc;
      }

      public byte[] getMsgBytes()
      {
            return msg;
      }
}



/**
  * @author: Mbonisi Masilela
  * File: LogWindow.java
  */

import javax.swing.*;
import java.awt.*;

public class LogWindow extends JFrame
{
      JTextArea area;
      JScrollPane pane;

      public LogWindow(String s)
      {
            super(s);
```

المنارة للاستشارات

www.manaraa.com

```
            Container c = getContentPane();
            c.setLayout(new BorderLayout());

            area = new JTextArea();

            c.add(new JScrollPane(area),
BorderLayout.CENTER);

            setSize(500, 400);
            setVisible(true);
            area.setEditable(false);
      }

      public void append(String s)
      {
            area.append(s);
            area.setCaretPosition(area.getText().length());
      }

      public String byteArrayToHexString(byte in[])
      {
            return BitUtils.byteArrayToHexString(in);
      }
}
```

المنارة للاستشارات
www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: Int2Byte.java
  */

public class Int2Byte
{
     public static void main(String m[])
     {
          byte[] t = new byte[2];
          t[0] = 0x3b;
          t[1] = 0xb;

          System.out.println("CRC = " +
BitUtils.byteArrayToInt(t));

          System.exit(0);

     }

     static String byteArrayToHexString(byte in[])
     {
          byte ch = 0x00;
          int i = 0;
          if (in == null || in.length <= 0)
               return null;

          String[] pseudo = {"0", "1", "2","3", "4", "5",
"6", "7",
                                   "8", "9", "A", "B", "C",
"D", "E","F"};
          StringBuffer out = new StringBuffer(in.length *
2);

          while (i < in.length)
          {
               ch = (byte) (in[i] & 0xF0); // Strip off
high nibble
               ch = (byte) (ch >>> 4);
                    // shift the bits down
               ch = (byte) (ch & 0x0F);
                    // must do this is high order bit is
on!
```

www.manaraa.com

```
                out.append(pseudo[ (int) ch]); // convert
the nibble to a String Character
                ch = (byte) (in[i] & 0x0F); // Strip off low
nibble
                out.append(pseudo[ (int) ch]); // convert
the nibble to a String Character
                out.append(" ");
                i++;
            }

            String rslt = new String(out);
            return rslt;
       }

     public static int byteArrayToInt(byte[] b)
     {
            int offset = 0;
          int value = 0;
          for (int i = 0; i < b.length; i++)
          {
              int shift = (b.length - 1 - i) * 8;
              value += (b[i + offset] & 0x000000FF) << shift;
          }
          return value;
       }

      public static byte[] intToByteArray(int val)
      {
            byte[] mybyte = new byte[4];

            mybyte[0] =(byte)( val >> 24 );
            mybyte[1] =(byte)( (val << 8) >> 24 );
            mybyte[2] =(byte)( (val << 16) >> 24 );
            mybyte[3] =(byte)( (val << 24) >> 24 );

            return mybyte;
      }

      public static byte inttobyte(int val)
      {
            return (byte)((val << 24) >> 24 );
      }

      public static int bytetoint(byte b)
```

المنارة للاستشارات

www.manaraa.com

```
	{
		return (int)(b & 0xFF);
	}

	static String byteToHexString(byte in)
	{
		byte ch = 0x00;
		int i = 0;

		String[] pseudo = {"0", "1", "2","3", "4", "5",
"6", "7",
					"8", "9", "A", "B", "C",
"D", "E","F"};

		StringBuffer out = new StringBuffer(2);

		ch = (byte) (in & 0xF0); // Strip off high nibble
		ch = (byte) (ch >>> 4);
		// shift the bits down
		ch = (byte) (ch & 0x0F);
		// must do this is high order bit is on!
		out.append(pseudo[ (int) ch]); // convert the
nibble to a String Character
		ch = (byte) (in & 0x0F); // Strip off low nibble
		out.append(pseudo[ (int) ch]); // convert the
nibble to a String Character
		out.append(" ");

		String rslt = new String(out);
		return rslt;
	}

}
```

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: FTSFile.java
  */

import java.io.*;
import java.util.*;

//File Transfer Send File class

public class FTSFile extends File
{
     int packetID;
     FileMsg[] packets;
     int numofpackets = 0;
     FileInputStream fis;
     String sendFileName;
     byte SRC_ADDR, DEST_ADDR;

     public FTSFile(String location, byte SRC, byte DEST)
throws Exception
     {
          super(location);

          System.out.println("Absolute Path "+
getAbsolutePath());
          this.packetID =
(int)(Calendar.getInstance().getTimeInMillis());

          this.DEST_ADDR = DEST;
          this.SRC_ADDR = SRC;

          //Check if file exists before proceeding with
normal operations
          try
          {
               fis = new FileInputStream(FTSFile.this);
          }
          catch(Exception e)
          {
               throw new Exception("Failed to open file " +
getAbsolutePath());
          }
```

المنارة للاستشارات

www.manaraa.com

```
            createSendFileName();
            createPackets();
      }

      private void createPackets()
      {
            packets = new FileMsg[getNumberOfPackets()];
            FileMsg tempPacket = new FileMsg();

            byte[] tempData = new
byte[FTConstants.DATA_MSG_SIZE];
            int flag = 0;

            for(int i = 0; i < packets.length; i++)
            {
                  tempData = new
byte[FTConstants.DATA_MSG_SIZE];
                  tempPacket=new FileMsg();
            //creates an empty packet
                  try
                  {
                        if(fis.available() <
FTConstants.DATA_MSG_SIZE)

      tempPacket.setLength(fis.available());
                        else

      tempPacket.setLength(FTConstants.DATA_MSG_SIZE);

                        flag = fis.read(tempData); //Read
FTConstants.DATA_MSG_SIZE or available() bytes
                  }
                  catch(Exception e)
                  {
                        e.printStackTrace();
                  }

                  tempPacket.setDestAddr(DEST_ADDR);
                  tempPacket.setSrcAddr(SRC_ADDR);
                  tempPacket.setData(tempData);
      //sets the data field of the packet
                  tempPacket.setMsgID(packetID);
      //sets the message id to packet
```

المنارة للاستشارات

www.manaraa.com

```
		tempPacket.setMsgType(FTConstants.FILE_DATA);
			//sets the message
				tempPacket.setPacketNumber(i);
			//sets the packet number
				packets[i] = tempPacket;
			//stores the temp packet into packet[]

				//If end of file is reached, print data from
	last packet
				/*if(flag==-1)
				{
					System.out.println("Error---");
					for (int j=0;j<tempData.length;j++)
					{

		System.out.println((char)tempData[j]);
					}
				}*/
			}

			try
			{
				fis.close();
			}
			catch(Exception e)
			{
			}
		}

		private void createSendFileName()
		{
			sendFileName = getName();

			if(sendFileName.length() >
	FTConstants.DATA_MSG_SIZE)
			{
				System.out.println("WARNING ::: The file
	name is too long .... Will be truncated to 20
	characters.");
				int dindex = sendFileName.lastIndexOf('.');
				if(dindex != -1)
				{
					if(dindex == 0)
```

المنارة للاستشارات

www.manaraa.com

```
                         sendFileName =
sendFileName.substring(0, FTConstants.DATA_MSG_SIZE - 2) +
"~";
                    else
                    {
                         String ext =
sendFileName.substring(dindex);
                         //very long file extention.
(probably not an extension)
                         if(ext.length() >
FTConstants.DATA_MSG_SIZE || ext.length() > 8)
                              sendFileName =
sendFileName.substring(0, FTConstants.DATA_MSG_SIZE - 2) +
"~";
                         else
                         {
                              int prefixSize =
FTConstants.DATA_MSG_SIZE - ext.length() - 1; //-1 for the
~ character to be placed
                              sendFileName =
sendFileName.substring(0, prefixSize) + "~" +
sendFileName.substring(dindex);
                         }
                    }
               }
               else
                    sendFileName =
sendFileName.substring(0, FTConstants.DATA_MSG_SIZE - 2) +
"~";
          }

          System.out.println("File name will be recieved as
" + sendFileName);
     }

     public FileMsg getPacketAt(int loc)
     {
          return packets[loc];
     }

     public int getNumberOfPackets()
     {
          int numberofpackets = (int)length() /
FTConstants.DATA_MSG_SIZE;
```

المنارة للاستشارات
www.manaraa.com

```
		if(length() %  FTConstants.DATA_MSG_SIZE != 0)
			numberofpackets++;

		return numberofpackets;
	}

	public String getSendFileName()
	{
		return sendFileName;
	}

	public int getPacketID()
	{
		return packetID;
	}
}
```

المنارة للاستشارات

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: BitUtils.java
  */

import java.io.*;
import java.util.*;

//File Transfer Receive File class

public class FTRFile  extends File
{
     private int packetID;
     FileMsg[] packets;
     int numofpackets = 0;
     FileOutputStream fos;
     String sendFileName;
     int packetCount;
     long starttime = 0;
     long stoptime = 0;
     byte SRC;

     public FTRFile(String location, byte src)throws
Exception
     {
          super(location);
          this.SRC = src;
          packetCount = 0;
          createSendFileName();
          starttime = (new Date()).getTime();
          //Just add a sec in case someone tries something
          stoptime = (new Date()).getTime() + 1000;
     }

     public FileMsg getPacketAt(int loc)
     {
          return packets[loc];
     }

     private void createSendFileName()
     {
          sendFileName = getName();
```

المنارة للاستشارات
www.manaraa.com

```
            if(sendFileName.length() >
FTConstants.DATA_MSG_SIZE)
            {
                  System.out.println("WARNING ::: The file
name is too long .... Will be truncated to 20
characters.");
                  int dindex = sendFileName.lastIndexOf('.');
                  if(dindex != -1)
                  {
                        if(dindex == 0)
                              sendFileName =
sendFileName.substring(0, FTConstants.DATA_MSG_SIZE - 2) +
"~";
                        else
                        {
                              String ext =
sendFileName.substring(dindex);
                              //very long file extention.
(probably not an extension)
                              if(ext.length() >
FTConstants.DATA_MSG_SIZE || ext.length() > 8)
                                    sendFileName =
sendFileName.substring(0, FTConstants.DATA_MSG_SIZE - 2) +
"~";
                              else
                              {
                                    int prefixSize =
FTConstants.DATA_MSG_SIZE - ext.length() - 1; //-1 for the
~ character to be placed
                                    sendFileName =
sendFileName.substring(0, prefixSize) + "~" +
sendFileName.substring(dindex);
                              }
                        }
                  }
                  else
                        sendFileName =
sendFileName.substring(0, FTConstants.DATA_MSG_SIZE - 2) +
"~";
            }

            System.out.println("File name will be recieved as
" + sendFileName);
      }
```

المنارة للاستشارات
www.manaraa.com

```
public byte getSrcAddr()
{
     return SRC;
}

public void addPacket(FileMsg packet)
{
     packets[packet.getPacketNumber()] = packet;
     packetCount++;
}


public int getNumberOfPackets()
{
     return numofpackets;
}

public void setNumberOfPackets(int numberOfPackets)
{
     packets = new FileMsg[numberOfPackets];
     numofpackets=numberOfPackets;
}

public int getPacketID()
{
     return packetID;
}

public void setPacketID(int id)
{
     packetID=id;
}


//Will write the file to disk
public void writeFile() throws Exception
{
     stoptime = (new Date()).getTime();

     FileOutputStream fos;
     //System.out.println(sendFileName);
     fos = new FileOutputStream(getAbsolutePath());
```

المنارة للاستشارات
www.manaraa.com

```
            //Goes through all packets except last one, and
write it to the output stream
            for (int i = 0; i < numofpackets - 1; i++)
            {
                  //System.out.println("" + packetID + " :::
Writing packet " + (i + 1) + " ..... Data length = " +
((packets[i].get_data() != null) ?
packets[i].get_data().length : 0));
                  fos.write(packets[i].getData());
                  fos.flush();
            }

            FileMsg lastPacket = packets[numofpackets-1];
            //Go through the last packet and write each byte
one at a time
            for(int i = 0; i < lastPacket.getLength(); i++)
            {
                  fos.write(lastPacket.getData(),i,1);
                  fos.flush();
            }

            fos.close();
            System.out.println("" + packetID + " ::: Done
writing file. File closed.");
      }

      public int getPacketCount()
      {
            return packetCount;
      }

      public float getFileTransferTime()
      {
            return (float)(stoptime - starttime) /
(float)1000;
      }
}
```

المنارة للاستشارات

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: FTConstants.java
  */

public class FTConstants
{
     public final static byte FILE_CREATE = 1;
     public final static byte FILE_CREATE_ACK = 10;
     public final static byte FILE_DATA = 2;
     public final static byte FILE_CLOSE = 3;
     public final static byte FILE_ERROR = 4;
     public final static byte FILE_REREQUEST = 5;
     public final static byte FILE_EXIT = 9;

     public final static int PACKET_SIZE = 236;
     public final static int DATA_MSG_SIZE = 225;

     public final static int TOTAL_PACKET_LEN_INDEX = 0;
     public final static int SEQ_NUM_INDEX = 1;
     public final static int DEST_ADDR_INDEX = 2;
     public final static int SRC_ADDR_INDEX = 3;
     public final static int PAYLOAD_LEN_INDEX = 4;
     public final static int MSG_TYPE_INDEX = 5;
     public final static int PACKET_ID_INDEX = 6;
     public final static int PACKET_NUM_INDEX = 7;     //2
bytes
     public final static int DATA_START_INDEX = 9;
     public final static int CRC_INDEX = (PACKET_SIZE - 2);

     public final static int MAX_DELIVERY_ATTEMPTS = 10;
     public final static int DELIVERY_WAIT_TIME = 2;
}

/*
Bytes
     0 - DEST ADDR
     1 - SRC ADDR
     2 - PAYLOAD LEN
     3 - MSG TYPE
     4 - Packet ID
     5 - Packet Number
     6 - 230 Data (225 data byte payload)
*/
```

المنارة للاستشارات

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: FileTransferS.java
  */

import java.io.*;
import java.util.*;
import javax.comm.*;

import java.awt.*;

/*
Bytes
     0 - DEST ADDR
     1 - SRC ADDR
     2 - PAYLOAD LEN
     3 - MSG TYPE
     4 - Packet ID
     5 - Packet Number
     6 - 230 Data (225 data byte payload)
*/


public class FileTransferS extends Component implements
FileAgent
{
     final byte LOCAL_ADDR, DEST_ADDR;
     public FTSFile file;
     String fileName;
     boolean isDone = false;
     int PACKET_ID = -1;
     byte dest, src;
     Serial2Mote mote;
     FileMsg ackPacket;
     boolean ackNotReceived = true;

    public FileTransferS(String source, String name, String
destination)
    {
          fileName = name;
          dest = Byte.parseByte(destination);
          src = Byte.parseByte(source);
          LOCAL_ADDR = src;
          DEST_ADDR = dest;
```

المنارة للاستشارات

www.manaraa.com

```
            ackPacket = null;
            mote = new Serial2Mote(this);
      }

       public void start() throws Exception
       {
            //Check if file exists before proceeding with
normal operations
            try
            {
                 file = new FTSFile(fileName, LOCAL_ADDR,
DEST_ADDR);
            }
            catch(Exception e)
            {
                 //System.out.println("Failed to open file "
+ fileName);
                 e.printStackTrace();
                 System.exit(0);
            }

            System.out.println("Sending file " + fileName + "
(" + file.length() + " bytes)");


            ackPacket = new FileMsg();

       ackPacket.setData(file.getSendFileName().getBytes());

       ackPacket.setLength(file.getSendFileName().length());
            ackPacket.setSrcAddr(LOCAL_ADDR);
            ackPacket.setDestAddr(DEST_ADDR);
            ackPacket.setMsgID(file.getPacketID());
            PACKET_ID = ackPacket.getMsgID();
            System.out.println("Packet ID = " + PACKET_ID);
            ackPacket.setMsgType(FTConstants.FILE_CREATE);

       ackPacket.setPacketNumber(file.getNumberOfPackets());

            //Send First Packet

            try
            {
                 int maxtries = 10;
```

المنارة للاستشارات

www.manaraa.com

```
                int c = 0;
                System.out.println("Waiting to ACK from DEST
= " + ackPacket.getDestAddr());
                //Keep resending
                while(ackNotReceived &&
FTConstants.MAX_DELIVERY_ATTEMPTS > c)
                {
                     mote.send(ackPacket);

     Thread.currentThread().sleep(FTConstants.DELIVERY_WAIT
_TIME * 1000);
                     c++;
                }

                if(ackNotReceived)
                {
                     System.out.println("Failed to reach
host " + ackPacket.getDestAddr() + " in " +
FTConstants.MAX_DELIVERY_ATTEMPTS + " attempts ....
exiting");
                     System.exit(0);
                }
           }
           catch(Exception e)
           {
                e.printStackTrace();
           }
     }

     public void sendRemainingPackets()
     {
           System.out.println("Preparing to send " +
file.getNumberOfPackets() + " data packets");

           long starttime = (new Date()).getTime();

           for(int i=0;i<file.packets.length;i++)
           {
                try
                {
                     mote.send(file.getPacketAt(i));
                     System.out.println("Sent Packet " + i);
                }
                catch(Exception e)
```

المنارة للاستشارات

www.manaraa.com

```
                {
                    e.printStackTrace();
                }
            }

            long stoptime = (new Date()).getTime();

            //Send final packet
            FileMsg packet = new FileMsg();
            packet.setMsgID(file.getPacketID());
            packet.setMsgType(FTConstants.FILE_CLOSE);
            packet.setPacketNumber(0);
            packet.setSrcAddr(LOCAL_ADDR);
            packet.setDestAddr(DEST_ADDR);
            float transferTime = (float)(stoptime -
starttime) / (float)1000;
            String timeString = "" + transferTime;
            packet.setData(timeString.getBytes());
            packet.setLength(timeString.length());

            try
            {
                mote.send(packet);
            }
            catch(Exception e){System.out.println("Failed to
send the close file packet ... abort
program");e.printStackTrace();}

            System.out.println("File transfer of " +
file.getSendFileName() + " is complete (" + file.length() +
" bytes sent). Packet Dispatch Time (" + (transferTime)+ "
seconds)\n");

            System.out.print("Waiting for shutdown approval
....... ");

        }

    public void messageReceived(FileMsg fmsg)
        {
            if(fmsg.getDestAddr() == LOCAL_ADDR)
            {
                if(fmsg.getMsgType() ==
FTConstants.FILE_CREATE_ACK)
```

www.manaraa.com

```
                {
                        if(fmsg.getPacketNumber() ==
ackPacket.getPacketNumber())
                        {
                                sendRemainingPackets();
                                ackNotReceived = false;
                        }
                        else
                        {
                                //Must have received a corrupt
packet ... resyn
                                try
                                {
                                        mote.send(ackPacket);
                                }
                                catch(Exception
e){System.out.println("Failed to resend the ack packet ...
abort program");e.printStackTrace();}
                        }
                }
                else if(fmsg.getMsgType() ==
FTConstants.FILE_REREQUEST)
                {
                        int index = fmsg.getPacketNumber();
                        System.out.print("Received Request to
resend packet " + index + " ..... ");
                        try
                        {

        mote.send(file.getPacketAt(index));
                                System.out.println("Resent packet
" + index);
                        }
                        catch(Exception e)
                        {
                                System.out.println("Failed to send
the rerequested packet");
                        }
                }
                else if(fmsg.getMsgType() ==
FTConstants.FILE_EXIT)
                {
                        if(fmsg.getMsgID() == PACKET_ID)
                        {
```

المنارة للاستشارات
www.manaraa.com

```
                         System.out.println("Shutdown
request received ... Exiting.");
                         System.out.println("File " +
file.getName() + " transfer successfully.\n");
                         System.exit(0);
                    }
               }
          }
     }

     public static void main(String[] args) throws
IOException
     {
          if(args.length != 3)
          {
               System.out.println("Please specify the
filename to be transfered.");
               System.exit(0);
          }

          try{(new FileTransferS(args[0], args[1],
args[2])).start();}catch(Exception e){}

          //System.exit(0);
     }
}
```

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: FileTransferR.java
  */


import java.io.*;
import java.util.*;

/*
Bytes
     0 - DEST ADDR
     1 - SRC ADDR
     2 - PAYLOAD LEN
     3 - MSG TYPE
     4 - Packet ID
     5 - Packet Number
     6 - 230 Data (225 data byte payload)
*/


public class FileTransferR implements FileAgent
{
     final byte LOCAL_ADDR;
     HashMap hashMap;

     final String OUTPUT_DIR_NAME = "RcvdFiles";
     String OUT_FILE_DIR = "";
     Serial2Mote mote;

     public FileTransferR(String addr)
     {
          LOCAL_ADDR = Byte.parseByte(addr);
          boolean outDirExists = true;

          if(!(new File(OUTPUT_DIR_NAME)).exists())
          {
               // Create a directory
               outDirExists = (new
File(OUTPUT_DIR_NAME)).mkdir();
               if(outDirExists)
                    OUT_FILE_DIR = OUTPUT_DIR_NAME + "/";
          }
          else
```

المنارة للاستشارات

www.manaraa.com

```
                OUT_FILE_DIR = OUTPUT_DIR_NAME + "/";

           System.out.println("All recieved files will be
stored " + (outDirExists ? "in the " + OUTPUT_DIR_NAME + "
directory." : " in the same directory as the program."));

           //Create Hash Table to keep track of received
fails
           hashMap = new HashMap();
           mote = new Serial2Mote(this);
      }

      public void messageReceived(FileMsg msg)
      {
           System.out.println("Checking local address .... "
+ msg.getDestAddr());
           if(msg.getDestAddr() == LOCAL_ADDR)
                processPacket(msg);
      }

      public void run()
      {
           while(true);
      }

      public void processPacket(FileMsg msg)
      {
           boolean err = false;

           switch(msg.getMsgType())
           {
                case FTConstants.FILE_CREATE:
                     try
                     {
                          String filename = OUT_FILE_DIR +
new String(msg.getData());
                          int id = msg.getMsgID();
                          FTRFile file = new
FTRFile(filename.trim(), msg.getSrcAddr());

      file.setNumberOfPackets(msg.getPacketNumber());
                          file.setPacketID(id);
```

www.manaraa.com

```
                                //create vector if it needs to be
created
                                String fileid = "" + id;

                                if(hashMap.get(fileid) == null)
                                {
                                     hashMap.put(fileid, file);
                                }

                                System.out.println("Recieved
FILE_CREATE request for " + file.getName() + ", creating
data structures. Expecting " + msg.getPacketNumber() + "
packets with ID " + msg.getMsgID());

                                FileMsg ackMsg = new FileMsg();
                                ackMsg.setMsgID(id);

     ackMsg.setDestAddr(msg.getSrcAddr());
                                ackMsg.setSrcAddr(LOCAL_ADDR);

     ackMsg.setMsgType(FTConstants.FILE_CREATE_ACK);

     ackMsg.setPacketNumber(msg.getPacketNumber());

                                mote.send(ackMsg);
                           }
                           catch(Exception e)
                           {
                                System.out.println("Failed to
create an output for packets with ID " + msg.getMsgID());
                                e.printStackTrace();
                                //System.exit(0);
                           }
                           break;
                      case FTConstants.FILE_DATA:
                           try
                           {
                                int id = msg.getMsgID();

                                //A data packet has been sent with
out no file create packet and
                                //no entry into the hash table at
this position has been made
                                String fileid = "" + id;
```

المنارة للاستشارات

www.manaraa.com

```
                         if(hashMap.get(fileid) == null)
                         {
                              System.out.println(fileid + "
::: No file has been created for this packet");
                              //System.exit(-1);  //not
sure if I want to exit
                         }

                         //Retrieves the file for this
packet from the hash table
                         FTRFile currentFile =
(FTRFile)hashMap.get(fileid);

                         //A data packet has been sent with
out no file create packet,
                         //there is an entry into the hash
table, just not this one
                         if(currentFile==null)
                         {
                              System.out.println(fileid + "
::: No file has been created for this packet");
                              //System.exit(0);  //not sure
if I want to exit
                         }

                         currentFile.addPacket(msg);

                         System.out.println(fileid + " :::
Received packet " + (msg.getPacketNumber() + 1) + " of "
                                                  +
currentFile.getNumberOfPackets() + " [Data Size = " +
msg.getData().length
                                                  + " ....
Supposed to be " + msg.getLength() + "]");
                    }

                    catch(Exception e)
                    {
                         System.out.println("" +
msg.getMsgID() + " ::: Failed to create an output for
packets with ID " + msg.getMsgID());
                         e.printStackTrace();
                         //System.exit(0);
                    }
```

المنارة للاستشارات
www.manaraa.com

```
                    break;
               case FTConstants.FILE_ERROR:
                    err = true;
               case FTConstants.FILE_CLOSE:
                    try
                    {
                         int id = msg.getMsgID();
                         //A file close packet has been
sent with out no file create packet and
                         //no entry into the hash table at
this position has been made
                         String fileid = "" + id;

                         (new FTRFileCloser(mote, hashMap,
fileid, LOCAL_ADDR)).start();
                    }
                    catch(Exception e)
                    {
                         System.out.println("" +
msg.getMsgID() + " ::: Error occurred for file with ID " +
msg.getMsgID());
                         e.printStackTrace();
                         //System.exit(0);
                    }
                    break;
          }
     }

     public static void main(String[] args) throws
IOException
     {
          if(args.length != 1)
               System.out.println("Please specify a valid
number of arguments ...");
          else
               (new FileTransferR(args[0])).run();
          System.exit(0);
     }

     class FTRFileCloser extends Thread implements Runnable
     {
          Serial2Mote mote;
          HashMap hashMap;
          String fileid;
```

المنارة للاستشارات
www.manaraa.com

```
            final int MAX_TRIES = 10;
            byte LOCAL_ADDR;

            public FTRFileCloser(Serial2Mote mote, HashMap
hashMap, String fileid, byte LOCAL_ADDR)
            {
                  this.mote = mote;
                  this.hashMap = hashMap;
                  this.fileid = fileid;
                  this.LOCAL_ADDR = LOCAL_ADDR;
            }

            public void run()
            {
                  boolean keepRunning = true;

                  try
                  {
                        //Maximum number of file rerequests
                        int numberoftries = 0;

                        //Retrieves the file for this packet
from the hash table
                        FTRFile currentFile =
(FTRFile)hashMap.get(fileid);

                        //A file close  packet has been sent
with out no file create packet,
                        //there is an entry into the hash
table, just not this one

                        if(currentFile==null)
                        {
                              System.out.println(fileid + " :::
No file has been created for this packet with ID" +
fileid);
                              throw new Exception("Packet data
structure not found .... discard the information.");
                        }

                        while(currentFile.getPacketCount() <
currentFile.getNumberOfPackets() && numberoftries <
MAX_TRIES)
                        {
```

www.manaraa.com

```
                         //request muissing packets
                         System.out.println(fileid + " :::
There are missing packets for file " +
currentFile.getName());
                         System.out.println(fileid + " :::
" + currentFile.getPacketCount()+" out of "+
currentFile.getNumberOfPackets()+
                                        " packets recieved
successfully.");
                         for(int
i=0;i<currentFile.getNumberOfPackets();i++)
                         {
                              //request missing packet

     if(currentFile.getPacketAt(i)==null)
                              {

     System.out.println(fileid + " ::: Rerequesting missing
packet "+ i);
                                   FileMsg rerequestPacket
= new FileMsg();
                                   //Empty data for a
rerequest packet

     //rerequestPacket.set_data(new byte[20]);

     rerequestPacket.setMsgID(currentFile.getPacketID());

     rerequestPacket.setMsgType(FTConstants.FILE_REREQUEST)
;

     rerequestPacket.setPacketNumber(i);

     rerequestPacket.setDestAddr(currentFile.getSrcAddr());

     rerequestPacket.setSrcAddr(LOCAL_ADDR);

     mote.send(rerequestPacket);
                              }
                         }

                         //Sleep and wait for a requested
packets to be sent back
```

المنارة للاستشارات
www.manaraa.com

```
                              int waittime =
currentFile.getNumberOfPackets() -
currentFile.getPacketCount();
                              System.out.println(fileid + " :::
Waiting for " + waittime + " seconds for rerequested
packets.");

     Thread.currentThread().sleep(waittime*1000);

                              numberoftries++;
                         }


                         if(numberoftries >= MAX_TRIES)
                         {
                              throw new Exception(fileid + " :::
Maxmimum number of data rerequests has been reached");
                         }
                         else
                         {
                              System.out.println(fileid + " :::
All Packets recieved were successfully received.");

                              currentFile.writeFile();

                              System.out.println(fileid + " :::
File " + currentFile.getName() + " has been created. All
transactions complete.");

                              FileMsg exitPacket = new
FileMsg();

     exitPacket.setMsgID(currentFile.getPacketID());

     exitPacket.setMsgType(FTConstants.FILE_EXIT);

     exitPacket.setDestAddr(currentFile.getSrcAddr());
                              exitPacket.setSrcAddr(LOCAL_ADDR);
                              mote.send(exitPacket);

                              System.out.println(fileid + " :::
File " + currentFile.getName() + " transfer completed in "
                                                       +
currentFile.getFileTransferTime() + " seconds. (" +
```

www.manaraa.com

```
(((float)currentFile.length() /
currentFile.getFileTransferTime()) / (float)1024) + "
Kbps)");
                    }
               }
               catch(Exception e)
               {
                    System.out.println(fileid + " ::: Error
occured while processing final file transactions for file
with ID " + fileid);
                    e.printStackTrace();
               }

               //Remove file from hashmap, all transactions
related to this file are complete
               hashMap.remove(fileid);
          }
     }
}
```

المنارة للاستشارات
www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: FTProtocolHandler.java
  */

public interface FTProtocolHandler
{
     public boolean isFileCreateAckReceived();
     public boolean isFileCloseAckReceived();
     public boolean isFileExitReceived();
     public boolean isFileErrorReceived();
     public boolean canSendNext();
     public void clearSendNext();
     public byte getSrcAddr();
     public byte getDestAddr();
}

/**
  * @author: Mbonisi Masilela
  * File: Snooper.java
  */


import java.util.*;
import javax.comm.*;
import java.io.*;

import java.awt.*;

public class Snooper extends Component
{
     Vector serialPortIDList = new Vector();
     SerialPort sport;
     OutputStream out;
     FileAgent agent;
     Vector msgVector;

     public final static String APP_TITLE = "Snooper";
     public final static int TIMEOUT_OPEN_SERIAL_PORT = 10;
     public final static int SECONDS = 1000;
     public final static int BAUD_RATE = 57600;

     public Snooper()
     {
```

المنارة للاستشارات

www.manaraa.com

```
          msgVector = new Vector();

          Enumeration portList =
CommPortIdentifier.getPortIdentifiers();
          while(portList.hasMoreElements())
          {
               CommPortIdentifier cpi =
(CommPortIdentifier)portList.nextElement();
               if(cpi.getPortType() ==
CommPortIdentifier.PORT_SERIAL)
                    serialPortIDList.add(cpi);
          }

          if(serialPortIDList.size() < 1)
               exit("Not enough COM ports where found. At
least 1 COM ports is required. (Found " +
serialPortIDList.size() + "!)");

          try
          {
               //COM1 is the top port but is always located
at index 0
               sport =
(SerialPort)((CommPortIdentifier)serialPortIDList.lastEleme
nt()).open(APP_TITLE, TIMEOUT_OPEN_SERIAL_PORT * SECONDS);
               //Create COMM Port Reader Daemons
               SerialPortReader portReader = new
SerialPortReader(sport);
               out = sport.getOutputStream();
               SerialPortWriter portWriter = new
SerialPortWriter(out, msgVector);

               (new Thread(portReader)).start();
               (new Thread(portWriter)).start();

               sport.setSerialPortParams(BAUD_RATE,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);

     sport.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);

               print("Listening on " + sport.getName() +
"\n");
          }
```

www.manaraa.com

```
		catch(Exception e)
		{
			e.printStackTrace();
			exit("Failed to open COM port for I/O.");
		}
	}

	public void run()
	{
		while(true);
	}

	public void send(FileMsg data) throws Exception
	{
		synchronized(msgVector)
		{
			msgVector.add(data);
		}
	}

	public void print(String s)
	{
		System.out.println(s);
	}

	public void exit(String s)
	{
		print(s);
		System.exit(0);
	}

	public static void main(String m[])
	{
		new Snooper();
	}

	class SerialPortWriter implements Runnable
	{
		OutputStream os;
		Vector msgVector;
		LogWindow outlog;

		public SerialPortWriter(OutputStream os, Vector
msgVector) throws Exception
```

المنارة للاستشارات
www.manaraa.com

```
          {
               this.os = os;
               this.msgVector = msgVector;
               outlog = new LogWindow("OUT LOG");
          }

          public void run()
          {
               while(true)
               {
                    FileMsg msg = null;
                    synchronized(msgVector)
                    {
                         if(msgVector.size() > 0)
                         {
                              msg =
(FileMsg)msgVector.elementAt(0);
                              msgVector.removeElementAt(0);
                         }
                    }
                    if(msg != null)
                    {
                         try
                         {

     outlog.append(BitUtils.byteArrayToHexString(msg.getMsg
Bytes()) + "\n\n");

     os.write(FTConstants.FRAME_DELIM);
                              os.write(msg.getMsgBytes());

     Thread.currentThread().sleep(msg.getMsgBytes().length)
;
                         }
                         catch(Exception e)
                         {
                              e.printStackTrace();
                         }
                    }
               }
          }
     }

     class SerialPortReader implements Runnable
```

المنارة للاستشارات
www.manaraa.com

```
	{
		SerialPort sp;
		BufferedReader commInput;
		InputStream is;
		FileAgent agent;
		LogWindow inlog;

		public final int FRAME_SIZE =
FTConstants.PACKET_SIZE;

		boolean startFrameFound, checkNext = false;
		int currIndex;
		int resets = 0;
		byte dataFrame[];

		public SerialPortReader(SerialPort sp) throws
Exception
		{
			this.sp = sp;
			inlog = new LogWindow("INPUT LOG");
			initCounters();
			resetSyncFlags();
			is = sp.getInputStream();
		}

		public void resetSyncFlags()
		{
			startFrameFound = false;
			checkNext = false;
		}

		public void initCounters()
		{
			startFrameFound = true;
			checkNext = false;
			currIndex = 0;
			dataFrame = new byte[FRAME_SIZE];
		}

		public void resetCounters()
		{
			currIndex = 0;
			startFrameFound = false;
			checkNext = false;
```

المنارة للاستشارات
www.manaraa.com

```
                dataFrame = new byte[FRAME_SIZE];
        }

        public void run()
        {
                while(true)
                {
                        try
                        {
                                int data;
                                while((data = is.read()) != -1)
                                {
                                        processData((byte)data);
                                }
                        }
                        catch(Exception e)
                        {
                                e.printStackTrace();
                        }
                }
        }

        public void processData(byte data)
        {
                if(!startFrameFound)
                {
                        if(!checkNext)
                        {
                                if(data ==
FTConstants.FRAME_DELIM)
                                {
                                        checkNext = true;
                                        //System.out.print("$#");
                                }
                                else
                                        System.out.println("Byte
thrown away for syncing - " + data);
                        }
                        else
                        {
                                if(data ==
BitUtils.inttobyte(FTConstants.PACKET_SIZE))
                                {
                                        initCounters();
```

المنارة للاستشارات
www.manaraa.com

```
                              dataFrame[currIndex] =
BitUtils.inttobyte(FTConstants.PACKET_SIZE);
                              currIndex++;
                         }
                         else if(data ==
FTConstants.FRAME_DELIM)
                         {
                              checkNext = true;
                              //System.out.print("$#");
                         }
                         else
                         {
                              System.out.println("Byte
thrown away for syncing - " + data);
                              resetSyncFlags();
                         }
                    }
               }
               else
               {
                    dataFrame[currIndex] = data;
                    //System.out.print(".");
                    if(currIndex == dataFrame.length - 1)
                    {
                         try
                         {

     inlog.append(BitUtils.byteArrayToHexString(dataFrame)
+ "\n\n");
                              FileMsg msg = new
FileMsg(dataFrame);
                              /*int msgCRC =
(short)BitUtils.byteArrayToInt(msg.getCRC());
                              int calcCRC =
BitUtils.calculateCRC(dataFrame);
                              if(msgCRC == calcCRC)

     agent.messageReceived(new FileMsg(dataFrame));
                              else
                              {

     System.out.println("################### INVALID PACKET
..... CORRUPT ::: CALC = " + calcCRC + " and MSG = " +
msgCRC);
```

المنارة للاستشارات
www.manaraa.com

```
                                        }*/
                                    }
                                    catch(Exception
e){System.out.println("FAILED TO CREATE FILE
MESSAGE");e.printStackTrace();}
                                    initCounters();
                                    resetSyncFlags();
                                    //System.out.print("#$");
                               }
                               else
                                    currIndex++;
                          }
                     }
                }
           }
```

المنارة للاستشارات

www.manaraa.com

```
/**
  * @author: Mbonisi Masilela
  * File: SerialSniffer.java
  */

import java.util.*;
import javax.comm.*;
import java.io.*;

public class SerialSniffer
{
     Vector serialPortIDList = new Vector();
     SerialPort sport;

     public final static String APP_TITLE = "Serial
Sniffer";
     public final static int TIMEOUT_OPEN_SERIAL_PORT = 10;
     public final static int SECONDS = 1000;
     public final static int BAUD_RATE = 57600;

     public static void main(String s[])
     {
          (new SerialSniffer()).run();
     }

     public SerialSniffer()
     {
          Enumeration portList =
CommPortIdentifier.getPortIdentifiers();
          while(portList.hasMoreElements())
          {
               CommPortIdentifier cpi =
(CommPortIdentifier)portList.nextElement();
               if(cpi.getPortType() ==
CommPortIdentifier.PORT_SERIAL)
                    serialPortIDList.add(cpi);
          }

          if(serialPortIDList.size() < 1)
               exit("Not enough COM ports where found. At
least 1 COM ports is required. (Found " +
serialPortIDList.size() + "!)");

          try
```

المنارة للاستشارات

www.manaraa.com

```
		{
			//always use last comm port
			sport = 
(SerialPort)((CommPortIdentifier)serialPortIDList.lastEleme
nt()).open(APP_TITLE, TIMEOUT_OPEN_SERIAL_PORT * SECONDS);
			sport.setSerialPortParams(BAUD_RATE, 
SerialPort.DATABITS_8, SerialPort.STOPBITS_1, 
SerialPort.PARITY_NONE);

			//Create COMM Port Reader Daemons
			SerialPortReader portReader = new 
SerialPortReader(sport);

			print("\nSerialSniffer: Listening on " + 
sport.getName() + "\n");
		}
		catch(Exception e)
		{
			e.printStackTrace();
			exit("Failed to open COM port for I/O.");
		}
	}

	public void run()
	{
		//Run forever
		while(true);
	}

	public void print(String s)
	{
		System.out.println(s);
	}

	public void exit(String s)
	{
		print(s);
		System.exit(0);
	}

	class SerialPortReader implements 
SerialPortEventListener
	{
		SerialPort sp;
```

www.manaraa.com

```
            BufferedReader commInput;
            InputStream is;

            public final int FRAME_SIZE = 39;
            public final byte FRAME_DELIM = 0x7E;

            //0x7D is the escape character ... remember to
support this

            boolean startFrameFound;
            boolean stopFrameFound;
            int currIndex;
            int resets = 0;
            byte dataFrame[];

            public SerialPortReader(SerialPort sp)
            {
                  this.sp = sp;

                  initCounters();

                  try
                  {
                        is = sp.getInputStream();
                        //commInput = new BufferedReader(new
InputStreamReader(sp.getInputStream()));
                        //Request to be notified when frames
arrive
                        sp.notifyOnDataAvailable(true);
                        //Set this class as the frame handler
                        sp.addEventListener(this);
                  }
                  catch(Exception e)
                  {
                        System.out.println("SerialPortReader
thread failed to open input stream to COM Port " +
sp.getName());
                        e.printStackTrace();
                        System.exit(0);
                  }
            }

            public void initCounters()
            {
```

www.manaraa.com

```
                currIndex = 0;
                startFrameFound = false;
                stopFrameFound = false;
                dataFrame = new byte[FRAME_SIZE];
        }

        public void serialEvent(SerialPortEvent event)
        {
                String inputString;

                //System.out.println("EVENT FIRED");

                switch(event.getEventType())
                {
                        case SerialPortEvent.BI:
                        case SerialPortEvent.OE:
                        case SerialPortEvent.FE:
                        case SerialPortEvent.PE:
                        case SerialPortEvent.CD:
                        case SerialPortEvent.CTS:
                        case SerialPortEvent.DSR:
                        case SerialPortEvent.RI:
                        case
SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                                break;
                        case SerialPortEvent.DATA_AVAILABLE:
                                try
                                {
                                        byte data[] = new
byte[is.available()];
                                        int bytesRead =
is.read(data);
                                        processData(data);
                                }
                                catch(Exception e)
                                {
                                        print("Error reading
packet");
                                        e.printStackTrace();
                                }
                                break;
                }
        }
```

www.manaraa.com

```
          public void firstFrameDelimFound(byte data)
          {
               initCounters();
               startFrameFound = true;
               dataFrame[currIndex] = data;
               currIndex++;

          }

          public void processData(byte[] data)
          {
               if(data[0] == 0x7E)
                    System.out.println();
               System.out.print(byteArrayToHexString(data)
+ " ");
          }

          String byteArrayToHexString(byte in[])
          {
          byte ch = 0x00;
          int i = 0;
          if (in == null || in.length <= 0)
               return null;

          String[] pseudo = {"0", "1", "2","3", "4", "5",
"6", "7",
                                       "8", "9", "A", "B", "C",
"D", "E","F"};
          StringBuffer out = new StringBuffer(in.length *
2);

          while (i < in.length)
          {
               ch = (byte) (in[i] & 0xF0); // Strip off
high nibble
               ch = (byte) (ch >>> 4);
                    // shift the bits down
               ch = (byte) (ch & 0x0F);
                    // must do this is high order bit is
on!
               out.append(pseudo[ (int) ch]); // convert
the nibble to a String Character
               ch = (byte) (in[i] & 0x0F); // Strip off low
nibble
```

www.manaraa.com

```
                out.append(pseudo[ (int) ch]); // convert
the nibble to a String Character
                out.append(" ");
                i++;
            }

            String rslt = new String(out);
            return rslt;
            }
        }
}



/**
  * @author: Mbonisi Masilela
  * File: Serial2Mote.java
  */


import java.util.*;
import javax.comm.*;
import java.io.*;

public class Serial2Mote
{
     Vector serialPortIDList = new Vector();
     SerialPort sport;
     OutputStream out;
     FileAgent agent;
     Vector msgVector;
     boolean showInLog = true;
     boolean showOutLog = true;

     public String APP_TITLE = "Serial2Mote";
     public final static int TIMEOUT_OPEN_SERIAL_PORT = 10;
     public final static int SECONDS = 1000;
     public final static int BAUD_RATE = 57600;

     SerialPortWriter portWriter;

     public Serial2Mote(FileAgent fileAgent, boolean
showin, boolean showout)
     {
```

المنارة للاستشارات
www.manaraa.com

```
		showInLog = showin;
		showOutLog = showout;
		this.agent = fileAgent;
		setup();
	}

	public Serial2Mote(FileAgent agent, boolean showLogs)
	{
		showInLog = showLogs;
		showOutLog = showLogs;
		this.agent = agent;
		setup();
	}

	public Serial2Mote(FileAgent fileAgent)
	{
		this.agent = fileAgent;
		setup();
	}

	public void setup()
	{
		msgVector = new Vector();

		Enumeration portList =
CommPortIdentifier.getPortIdentifiers();
		while(portList.hasMoreElements())
		{
			CommPortIdentifier cpi =
(CommPortIdentifier)portList.nextElement();
			if(cpi.getPortType() ==
CommPortIdentifier.PORT_SERIAL)
				serialPortIDList.add(cpi);
		}

		if(serialPortIDList.size() < 1)
			exit("Not enough COM ports where found. At
least 1 COM ports is required. (Found " +
serialPortIDList.size() + "!)");

		try
		{
			//COM1 is the top port but is always located
at index 0
```

المنارة للاستشارات

www.manaraa.com

```
                    sport =
(SerialPort)((CommPortIdentifier)serialPortIDList.firstElem
ent()).open(APP_TITLE, TIMEOUT_OPEN_SERIAL_PORT * SECONDS);
                    //Create COMM Port Reader Daemons
                    SerialPortReader portReader = new
SerialPortReader(agent, sport, showInLog);
                    out = sport.getOutputStream();
                    portWriter = new SerialPortWriter(out,
msgVector, showOutLog);

                    (new Thread(portReader)).start();
                    (new Thread(portWriter)).start();

                    sport.setSerialPortParams(BAUD_RATE,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);

      sport.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);

                    print("Listening on " + sport.getName() +
"\n");
             }
             catch(Exception e)
             {
                    e.printStackTrace();
                    exit("Failed to open COM port for I/O.");
             }
      }

      public void run()
      {
             while(true);
      }

      public void send(FileMsg data) throws Exception
      {
             synchronized(msgVector)
             {
                    msgVector.add(data);
             }
      }

      public void sendS(FileMsg data)
      {
```

المنارة للاستشارات
www.manaraa.com

```
			portWriter.send(data);
		}

		public void print(String s)
		{
			System.out.println(s);
		}

		public void exit(String s)
		{
			print(s);
			System.exit(0);
		}

		class SerialPortWriter implements Runnable
		{
			OutputStream os;
			Vector msgVector;
			LogWindow outlog;
			boolean showLog;

			public SerialPortWriter(OutputStream os, Vector
	msgVector, boolean showlog) throws Exception
			{
				this.os = os;
				this.msgVector = msgVector;
				this.showLog = showlog;
				if(showLog)
					outlog = new LogWindow("OUT LOG");
			}

			public void send(FileMsg msg)
			{
				try
				{
					os.write(FTConstants.FRAME_DELIM);
					os.write(msg.getMsgBytes());
					if(showLog)

		outlog.append(BitUtils.byteArrayToHexString(msg.getMsg
	Bytes()) + "\n\n");
					Thread.currentThread().sleep(100);
				}
				catch(Exception e)
```

المنارة للاستشارات
www.manaraa.com

```
                {
                        e.printStackTrace();
                }
        }

        public void run()
        {
                while(true)
                {
                        FileMsg msg = null;
                        synchronized(msgVector)
                        {
                                if(msgVector.size() > 0)
                                {
                                        msg =
(FileMsg)msgVector.elementAt(0);
                                        msgVector.removeElementAt(0);
                                }
                        }
                        if(msg != null)
                        {
                                try
                                {
                                        if(showLog)

        outlog.append(BitUtils.byteArrayToHexString(msg.getMsg
Bytes()) + "\n\n");

        os.write(FTConstants.FRAME_DELIM);
                                        os.write(msg.getMsgBytes());

        //Thread.currentThread().sleep(msg.getMsgBytes().lengt
h / 5);

                                }
                                catch(Exception e)
                                {
                                        e.printStackTrace();
                                }
                        }
                }
        }
    }
```

المنارة للاستشارات
www.manaraa.com

```
class SerialPortReader implements Runnable
{
      SerialPort sp;
      BufferedReader commInput;
      InputStream is;
      FileAgent agent;
      LogWindow inlog, synclog;

      public final int FRAME_SIZE =
FTConstants.PACKET_SIZE;

      boolean startFrameFound, checkNext = false;
      int currIndex, syncIndex;
      int resets = 0;
      byte dataFrame[];
      String partialSyncData = "";
      String snl = "";
      boolean showLogs;

      public SerialPortReader(FileAgent agent,
SerialPort sp, boolean showlogs) throws Exception
      {
            this.agent = agent;
            this.sp = sp;
            this.showLogs = showlogs;

            if(showLogs)
            {
                  inlog = new LogWindow("INPUT LOG");
                  synclog = new LogWindow("SYNC
PACKETS");
            }
            initCounters();
            resetSyncFlags();
            is = sp.getInputStream();
      }

      public void resetSyncFlags()
      {
            startFrameFound = false;
            checkNext = false;
      }

      public void initCounters()
```

www.manaraa.com

```
{
	startFrameFound = true;
	checkNext = false;
	currIndex = 0;
	dataFrame = new byte[FRAME_SIZE];
	resetSyncInfo();
}

public void resetSyncInfo()
{
	syncIndex = 0;
	partialSyncData = "";
}

public void resetCounters()
{
	currIndex = 0;
	startFrameFound = false;
	checkNext = false;
	dataFrame = new byte[FRAME_SIZE];
}

public void run()
{
	while(true)
	{
		try
		{
			int data;
			while((data = is.read()) != -1)
			{
				processData((byte)data);
			}
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}

public void trackSyncData(byte data)
{
```

www.manaraa.com

```
                    //System.out.println("Byte thrown away for
syncing - " + data);
                    partialSyncData +=
BitUtils.byteToHexString(data);

                    if(syncIndex == FRAME_SIZE - 1)
                    {
                         if(showLogs)
                              synclog.append("FULL SYNC DATA\n"
+ partialSyncData + "\n\n");
                         resetSyncInfo();
                    }
                    else
                         syncIndex++;
                    resetSyncFlags();
               }

               public void processData(byte data)
               {
                    if(!startFrameFound)
                    {
                         if(!checkNext)
                         {
                              if(data ==
FTConstants.FRAME_DELIM)
                              {
                                   checkNext = true;
                                   //System.out.print("$#");
                              }
                              else
                                   trackSyncData(data);
                         }
                         else
                         {
                              if(data ==
BitUtils.inttobyte(FTConstants.PACKET_SIZE))
                              {
                                   if(syncIndex > 0 && showLogs)
                                        synclog.append("PARTIAL
SYNC DATA\n" + partialSyncData + "\n\n");
                                   initCounters();
                                   dataFrame[currIndex] =
BitUtils.inttobyte(FTConstants.PACKET_SIZE);
                                   currIndex++;
```

المنارة للاستشارات
www.manaraa.com

```
                         }
                         else if(data ==
FTConstants.FRAME_DELIM)
                         {
                              checkNext = true;
                              //System.out.print("$#");
                         }
                         else
                         {
                              trackSyncData(data);
                         }
                    }
               }
               else
               {
                    dataFrame[currIndex] = data;
                    //System.out.print(".");
                    if(currIndex == dataFrame.length - 1)
                    {
                         try
                         {
                              FileMsg msg = new
FileMsg(dataFrame);
                              int msgCRC =
(short)BitUtils.byteArrayToInt(msg.getCRC());
                              int calcCRC =
BitUtils.calculateCRC(dataFrame);
                              if(msgCRC != calcCRC &&
showLogs)
                                   inlog.append("####
CORRUPT PACKET ::: CALCULATED CRC = " + calcCRC + " and MSG
CRC = " + msgCRC + "\n");
                              else

     agent.messageReceived(new FileMsg(dataFrame));

                              if(showLogs)

     inlog.append(BitUtils.byteArrayToHexString(dataFrame)
+ "\n\n");
                         }
                         catch(Exception
e){System.out.println("FAILED TO CREATE FILE
MESSAGE");e.printStackTrace();}
```

المنارة للاستشارات

www.manaraa.com

```
                        initCounters();
                        resetSyncFlags();
                        //System.out.print("#$");
                    }
                    else
                        currIndex++;
                }
            }
        }
}



/**
  * @author: Mbonisi Masilela
  * File: PacketInjector.java
  */

import javax.swing.*;
import java.awt.*;

public class PacketInjector extends JFrame
{
     public PacketInjector(String s)
     {
          super(s);
     }
}

class LTF extends JPanel
{
     JLabel label;
     JTextField field;

     public LTF(String labelText)
     {
          super();
          setLayout(new GridLayout(1,2));
          label = new JLabel(labelText);
          field = new JTextField();
          add(label);
          add(field);
     }
```

المنارة للاستشارات

www.manaraa.com

```
    public void setLabelText(String s)
    {
        label.setText(s);
    }

    public void setInputText(String s)
    {
        field.setText(s);
    }

    public String getInputText()
    {
        return field.getText();
    }
}
```

للاستشارات المنارة

www.manaraa.com