DYNAMIC ROUTING FOR DATA INTEGRITY AND DELAY DIFFERENTIATED SERVICES IN WIRELESS SENSOR NETWORKS

[1]PATHREE RAGHU SARAN, [2]O.KOTESHWAR RAO

[1]M.Tech student, [2]Associate Professor
[1]Computer Science and Engineering,
Brahmaiah College of Engineering, North rajupalem, Nellore,Andhra Pradesh, India.

***Abstract*: Applications running on the same Wireless Sensor Network (WSN) platform usually have different Quality of Service (QoS) requirements. Two basic requirements are low delay and high data integrity. However, in most situations, these two requirements cannot be satisfied simultaneously. Based on the concept of potential in physics, IDDR is proposed, which is a multi-path dynamic routing algorithm, to resolve this conflict. By constructing a virtual hybrid potential field, IDDR separates packets of applications with different QoS requirements according to the weight assigned to each packet, and routes them towards the sink through different paths to improve the data fidelity for integrity-sensitive applications as well as reduce the end-to-end delay for delay-sensitive ones.**

***Index Terms*: Component, formatting, style, styling, insert**

I INTRODUCTION

WSNS, which are used to sense the physical world, will play an important role in the next generation networks. Due to the diversity and complexity of applications running over WSNs, the QoS guarantee in such networks gains increasing attention in the research community. As a part of an information infrastructure, WSNs should be able to support various applications over the same platform. Different applications might have different QoS requirements. For instance, in a fire monitoring application, the event of a fire alarm should be reported to the sink as soon as possible. On the other hand, some applications require most of their packets to successfully arrive at the sink irrespective of when they arrive. For example, in habitat monitoring applications, the arrival of packets is allowed to have a delay, but the sink should receive most of the packets. WSNs have two basic QoS requirements: low delay and high data integrity, leading to what are called delay sensitive applications and high-integrity applications, respectively.
Generally, in a network with light load, both requirements can be readily satisfied. However, a heavily loaded network will suffer congestion, which increases the end-to-end delay.

OBJECTIVES OF STUDY

To improve the fidelity for high-integrity applications and decrease the end-to-end delay for delay-sensitive ones, even when the network is congested.
The basic idea is to find as much buffer space as possible from the idle and/or under-loaded paths to cache the excessive packets that might be dropped on the shortest path.

II EXISTING SYSTEM

Most QoS provisioning protocols proposed for traditional ad hoc networks have large overhead caused by end-to-end path discovery and resource reservation.
They are not suitable for resource-constrained WSNs.
Providing Real-Time Service
RAP exploits the notion of velocity and proposes a velocity monotonic scheduling policy to minimize the ratio of missed deadlines. However, the global information of network topology is required.
Implicit Earliest Deadline First (EDF) mainly utilizes a medium access control protocol to provide real-time service. The implicit prioritization is used instead of relying on control packets as most other protocols do.
SPEED maintains a desired delivery speed across the network through a novel combination of feedback control and non-deterministic QoS-aware geographic forwarding.
A two-hop neighbor information-based gradient routing mechanism is proposed to enhance realtime performance. The routing decision is made based on the number of hops from a source to the sink and the twohop information.

Providing Reliability Service
Adaptive Forwarding Scheme (AFS) employs the packet priority to determine the forwarding behavior to control the reliability.
ReInforM uses the concept of dynamic packet states to control the number of paths required for the desired reliability.
However, both of AFS and ReInforM require to know the global network topology.
LIEMRO utilizes a dynamic path maintenance mechanism to monitor the quality of the active paths during network operation and regulates the injected traffic rate of the paths according to the latest perceived paths quality.

However, it does not consider the effects of buffer capacity and service rate of the active nodes to estimate and adjust the traffic rate of the active paths.

Providing Real-Time and Reliability Services

MMSPEED extends SPEED for service differentiation and probabilistic QoS guarantee. It uses the same mechanism as SPEED to satisfy the delay requirements for different types of traffic, and uses redundant paths to ensure reliability. The MAC layer function is modified to provide prioritized access and reliable multicast delivery of packets to multiple neighbors.

However, when the network is congested, all the source nodes still continuously transmit packets to the sink along multipaths without taking some other mechanisms, such as caching packets for some time.

Energy-Efficient and QoS based Multipath Routing Protocol (EQSR) improves reliability through using a lightweight XOR-based Forward Error Correction (FEC) mechanism, which introduces data redundancy in the data transmission process.

III PROPOSED SYSTEM

First task is to find these idle and/or under loaded paths, then the second task is to cache the packets efficiently for subsequent transmission.

IDDR constructs a potential field according to the depth1 and queue length information to find the under-utilized paths.

The packets with high integrity requirement will be forwarded to the next hop with smaller queue length. A mechanism called Implicit Hop-by-Hop Rate Control is designed to make packet caching more efficient and decrease end-to-end delay for delay-sensitive applications.

Each application is assigned a weight, which represents the degree of sensitivity to the delay. Through building local dynamic potential fields with different slopes according to the weight values carried by packets, IDDR allows the packets with larger weight to choose shorter paths. In addition, IDDR also employs the priority queue to further decrease the queuing delay of delay-sensitive packets.

IDDR inherently avoids the conflict between high integrity and low delay: the high-integrity packets are cached on the under loaded paths along which packets will suffer a large end-to-end delay because of more hops, and the delay-sensitive packets travel along shorter paths to approach the sink as soon as possible.

Advantages

Packet caching more efficient

Decrease end-to-end delay for delay-sensitive applications

HARDWARE REQUIREMENTS

Processor : Any Processor above 500 MHz.
Ram : 128Mb.
Hard Disk : 10 Gb.
Compact Disk : 650 Mb.
Input device : Standard Keyboard and Mouse.
Output device : VGA and High Resolution Monitor.

SOFTWARE SPECIFICATION

Operating System : Windows Family.
Techniques : JDK 1.5 or higher
Database : MySQL 5.0

IV SOFTWARE DESCRIPTION:

JAVA

Java is an object-oriented multithread programming languages .It is designed to be small, simple and portable across different platforms as well as operating systems.

FEATURES OF JAVA

Platform Independence

The Write-Once-Run-Anywhere ideal has not been achieved (tuning for different platforms usually required), but closer than with other languages.

Object Oriented

Object oriented throughout - no coding outside of class definitions, including main().

An extensive class library available in the core language packages.

Compiler/Interpreter Combo

Code is compiled to byte codes that are interpreted by a Java virtual machines (JVM).

This provides portability to any machine for which a virtual machine has been written.

The two steps of compilation and interpretation allow for extensive code checking and improved security.

Robust

Exception handling built-in, strong type checking (that is, all data must be declared an explicit type), local variables must be initialized.

Several features of C & C++ eliminated:

No memory pointers

No preprocessor

Array index limit checking

Automatic Memory Management
Automatic garbage collection - memory management handled by JVM.
Security
No memory pointers
Programs run inside the virtual machine sandbox.
Array index limit checking
Code pathologies reduced by
Byte code verifier - checks classes after loading
Class loader - confines objects to unique namespaces. Prevents loading a hacked "java.lang.SecurityManager" class, for example.
Security manager - determines what resources a class can access such as reading and writing to the local disk.
Dynamic Binding
The linking of data and methods to where they are located is done at run-time.
New classes can be loaded while a program is running. Linking is done on the fly.
Even if libraries are recompiled, there is no need to recompile code that uses classes in those libraries.
This differs from C++, which uses static binding. This can result in fragile classes for cases where linked code is changed and memory pointers then point to the wrong addresses.

Good Performance
Interpretation of byte codes slowed performance in early versions, but advanced virtual machines with adaptive and just-in-time compilation and other techniques now typically provide performance up to 50% to 100% the speed of C++ programs.
Threading
Lightweight processes, called threads, can easily be spun off to perform multiprocessing.
Can take advantage of multiprocessors where available
Great for multimedia displays.
Built-in Networking
Java was designed with networking in mind and comes with many classes to develop sophisticated Internet communications.
IMP applications are called IMlets, but in reality they are MIDlets. They subclass MIDlet, and follow the same packaging, deployment, security and life-cycle as MIDlets.

Connected Device Configuration
CDC is a smaller subset of Java SE, containing almost all the libraries that are not GUI related.

Foundation Profile
A headless version of Java SE.

Personal Basis Profile
Extends the Foundation Profile to include lightweight GUI support in the form of an AWT subset.

Personal Profile
This extension of Personal Basis Profile includes a more comprehensive AWT subset and adds applet support.
The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, Drupal and other software built on the LAMP software stack.

Features
MySQL offered MySQL 5.1 in two different variants: the open source MySQL Community Server and the commercial Enterprise Server. MySQL 5.5 is offered under the same licences. They have a common code base and include the following features:

* A broad subset of ANSI SQL 99, as well as extensions
* Cross-platform support
* Stored procedures
* Triggers
* Cursors
* Updatable Views
* True Varchar support
* Information schema
* Strict mode[further explanation needed]

* X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using Oracle's InnoDB engine
* Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, MySQL Archive for storing historical data in little space)
* Transactions with the InnoDB, and Cluster storage engines; savepoints with InnoDB
* SSL support
* Query caching
* Sub-SELECTs (i.e. nested SELECTs)
* Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master, no automatic support for multiple masters per slave.
* Full-text indexing and searching using MyISAM engine
* Embedded database library
* Partial Unicode support (UTF-8 and UCS-2 encoded strings are limited to the BMP)
* ACID compliance when using transaction capable storage engines (InnoDB and Cluster)[28]
* Partititoned tables with pruning of partitions in optimiser
* Shared-nothing clustering through MySQL Cluster
* Hot backup (via mysqlhotcopy) under certain conditions

SYSTEM ARCHITECTURE

MODULES
Nodes
Route discovery
Delay sensitive routing
Integrity sensitive routing
Data delivery

Nodes
Sensor nodes are created with 'N' number of nodes. A commonly used routing algorithm will choose the optimal path for all the packets. Sensor nodes can transmit packets via delay sensitive and integrity sensitive.
Resource Discovery
In a under-utilized WSN, the queue length is very small, the hybrid potential field is governed by the depth potential field. IDDR performs like the shortest path algorithm, that is, a node always chooses one neighbor with lower depth as its next hop. However, in a over-utilized WSN, the shortest paths are likely be full of packets. Therefore, new coming packets will be driven out of the shortest paths to find other available resource. If a node knows the queue length information of its neighbors, it can forward packets to the underloaded neighbors to stand against possible dropping.

Delay sensitive routing
Delay-sensitive packets occupy the limited bandwidth and buffers, worsening drops of high-integrity ones. Delay sensitive packet forwarding takes shortest pacth or less hop relay path to transmit the packets.
Integrity sensitive routing
High-integrity packets block the shortest paths, compelling the delay-sensitive packets to travel more hops before reaching the sink, which increases the delay. High-integrity packets occupy the buffers, which also increases the queuing delay of delay-sensitive packets. To overcome the above drawbacks, a mechanism is designed, which allows the delay-sensitive packets to move along the shortest path and the packets with fidelity requirements to detour to avoid possible dropping on the hotspots. In this way, the data integrity and delay differentiated services can be provided in the same network. Integrity sensitive packet forwarding takes longest route, less traffic route to route the packets.

Data delivery
The basic idea of IDDR is to consider the whole network as a big buffer to cache the excessive packets before they arrive at the sink. Data from sensor nodes delivered to sink node via two different routing techniques such as delay sensitive and integrity sensitive. One of the options of these two can be chosen by user. Finally data from node arrive sink node. For security purpose, the data is encrypted at sensor nodes and sent to sink, whereas sink can decrypt and get the packets.

UML DIAGRAM
USE CASE DIAGRAM

Sample Code

Sink.java
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
import javax.swing.event.*;
import java.util.*;
import java.net.*;
import java.io.*;
import java.sql.*;
import java.text.DateFormat;
import java.util.Date.*;
import org.jvnet.substance.SubstanceLookAndFeel;

public class Sink extends JFrame
{

	private JTabbedPane jTabbedPane1;
	private JButton jButton1;
	private JButton jButton2;
	private JButton jButton3;
	private JPanel contentPane;


	private JTextArea jTextArea1;
	private JScrollPane jScrollPane4;
	private JPanel jPanel4;

	private JTextArea jTextArea2;
	private JScrollPane jScrollPane5;

	private JPanel jPanel5;

	private JTextArea jTextArea3;
	private JScrollPane jScrollPane6;
	private JPanel jPanel6;
	String name="sink";
	String str1;
  String str2;
	String str3;
	ServerSocket Sink_S1;
	ServerSocket Sink_S2;
	ServerSocket Sink_S3;
	Socket Sink_C;
	String msg="";
	StringBuffer sb;
	String stee="";
	int JT=0;
	String On_Childs="";
		String Off_Childs="";


	public Sink()
	{
		super();
		initializeComponent();
		try
		{
			Sink_S1=new ServerSocket(6661);

		}
		catch (Exception exp)
		{
			exp.printStackTrace();
		}

	this.setVisible(true);
```

```
}
private void initializeComponent()
{
	jTabbedPane1 = new JTabbedPane();
	jButton1 = new JButton();
	jButton2 = new JButton();
	jButton3= new JButton();
	contentPane = (JPanel)this.getContentPane();

	jTextArea1 = new JTextArea();

	jScrollPane4 = new JScrollPane();
	jPanel4 = new JPanel();

	jTextArea2 = new JTextArea();
	jScrollPane5 = new JScrollPane();
	jPanel5 = new JPanel();

	jTextArea3 = new JTextArea();
	jScrollPane6 = new JScrollPane();
	jPanel6 = new JPanel();

	jTextArea1.setFont(new Font("Serif",Font.BOLD,18));
	jTextArea2.setFont(new Font("Serif",Font.BOLD,18));
	jTextArea3.setFont(new Font("Serif",Font.BOLD,18));

	//jTabbedPane1.addTab("LP Sink1", jPanel4);
	jTabbedPane1.addTab("Integrity Sensitive", jPanel5);
	jTabbedPane1.addTab("Delay Sensitive", jPanel6);

	jTabbedPane1.addChangeListener(new ChangeListener() {
		public void stateChanged(ChangeEvent e)
		{
			jTabbedPane1_stateChanged(e);
		}

	});

	jButton1.setText("Save ");
	jButton1.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e)
		{
			jButton1_actionPerformed(e);
		}

	});

	jButton2.setText("Exit");
	jButton2.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e)
		{
			jButton2_actionPerformed(e);
		}

	});
	jButton3.setText("Clear");
	jButton3.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e)
		{
			jButton3_actionPerformed(e);
		}
```

```
        });

        contentPane.setLayout(null);
        contentPane.setBackground(new Color(204, 204, 255));
        addComponent(contentPane, jTabbedPane1, 15,7,515,363);
         jTabbedPane1.setBackground(new Color(204, 204, 255));
        addComponent(contentPane, jButton1, 117,388,88,32);
        addComponent(contentPane, jButton2, 227,388,88,32);
        addComponent(contentPane, jButton3, 347,388,88,32);

        jScrollPane4.setViewportView(jTextArea1);

        jPanel4.setLayout(null);
        addComponent(jPanel4, jScrollPane4, 6,6,496,322);

        jTextArea2.setText(" ");

        jScrollPane5.setViewportView(jTextArea2);

        jPanel5.setLayout(null);
        addComponent(jPanel5, jScrollPane5, 6,6,496,322);

        jTextArea3.setText(" ");

        jScrollPane6.setViewportView(jTextArea3);

        jPanel6.setLayout(null);
        addComponent(jPanel6, jScrollPane6, 6,6,496,322);

        this.setTitle("Sink - extends JFrame");
        this.setLocation(new Point(0, 0));
        this.setSize(new Dimension(556, 452));
}

private void addComponent(Container container,Component c,int x,int y,int width,int height)
{
        c.setBounds(x,y,width,height);
        container.add(c);
}

private void jTabbedPane1_stateChanged(ChangeEvent e)
{
        if (jTabbedPane1.getSelectedIndex()==0)
        {
                stee=jTextArea1.getText();

        }
        else if (jTabbedPane1.getSelectedIndex()==1)
        {
                stee=jTextArea2.getText();

        }
        else if (jTabbedPane1.getSelectedIndex()==2)
        {
                stee=jTextArea3.getText();

        }

}
```

```
private void jButton1_actionPerformed(ActionEvent e)
        {
                try
                {

                                        FileDialog fd=new FileDialog(this,"Save",FileDialog.SAVE);
                                        fd.show();

                                        FileOutputStream f=new FileOutputStream(fd.getDirectory()+fd.getFile());

                                        byte b[]=stee.getBytes();
                                        f.write(b);

                        Sink ss1=new Sink();
                        while(true)
                        {
                                ss1.Sink2S();
                        }
        }

}
```

SAMPLE SCREEN SHOTS

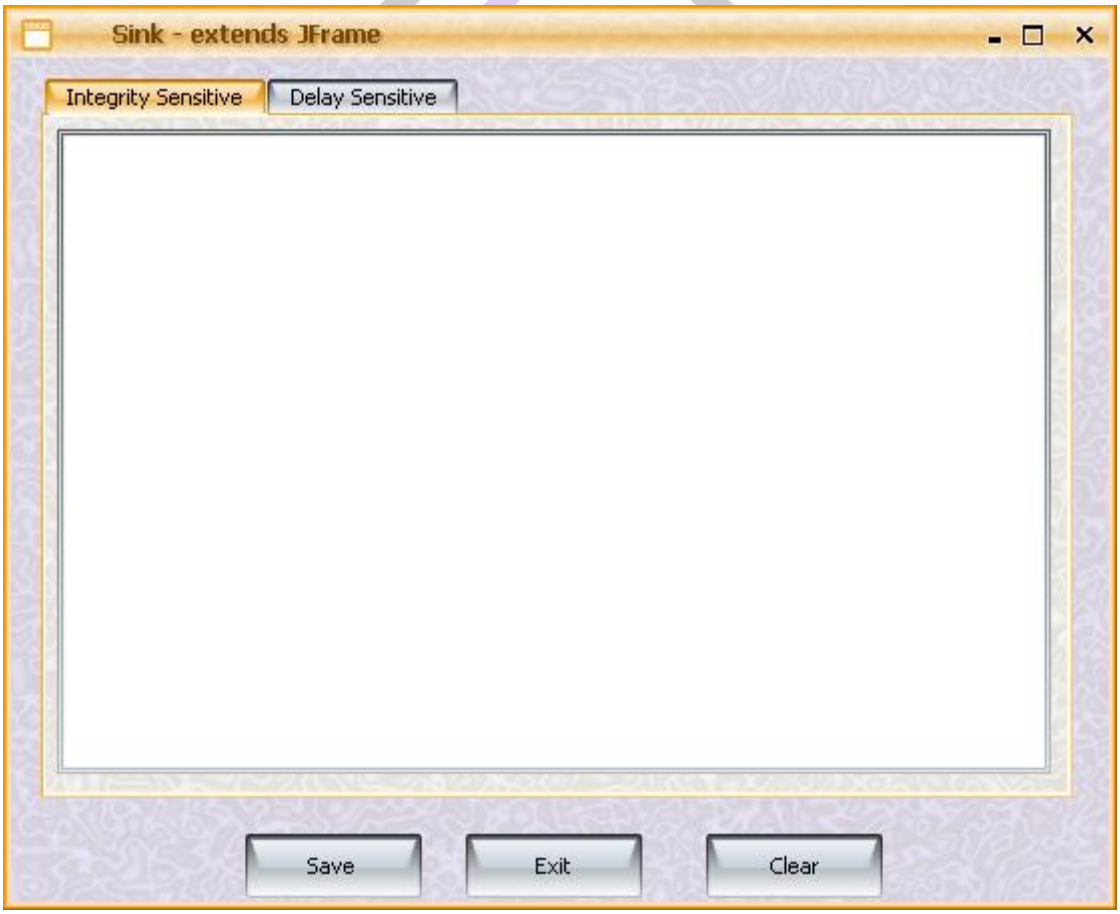

Figure 1 J Frame

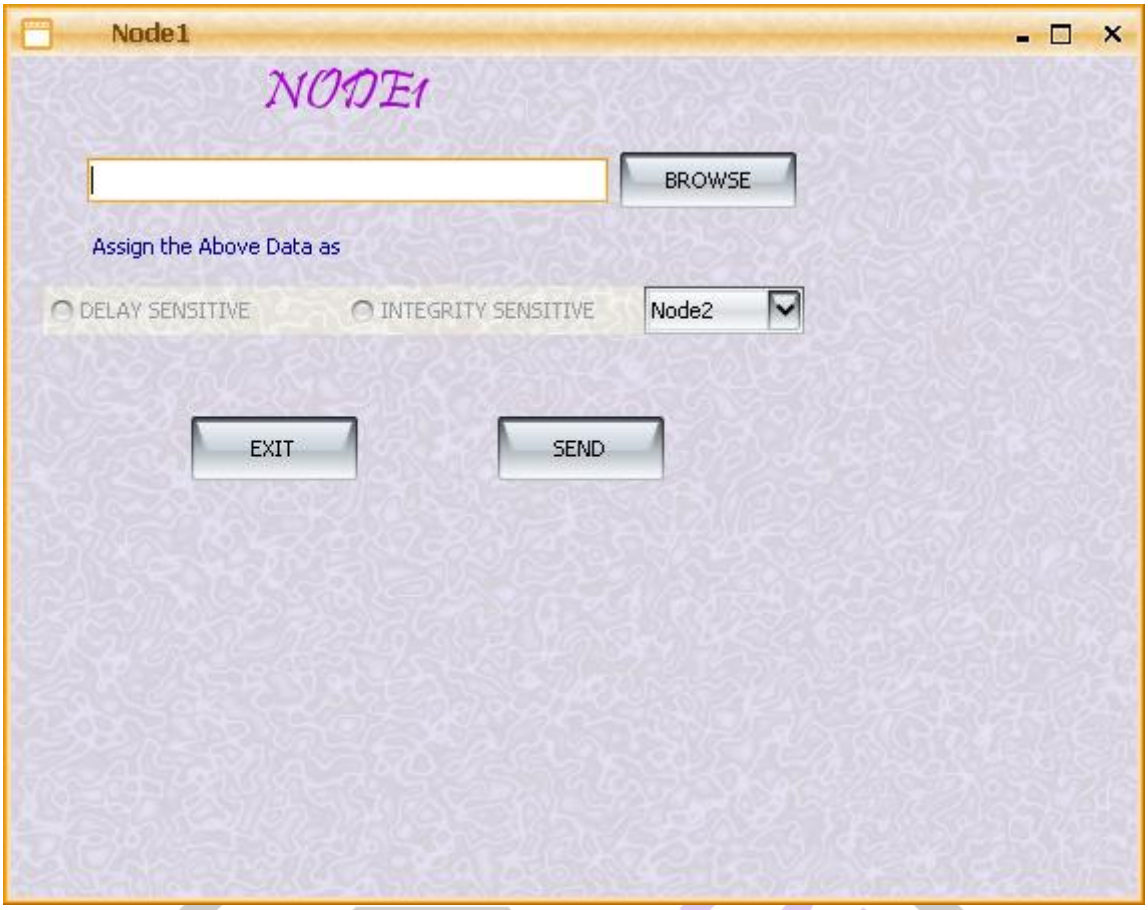

Figure 2 Node

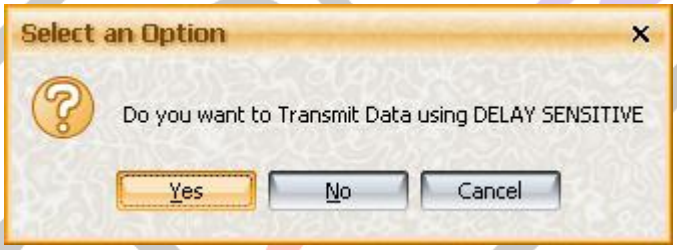

Figure 3 Delay Sensitive

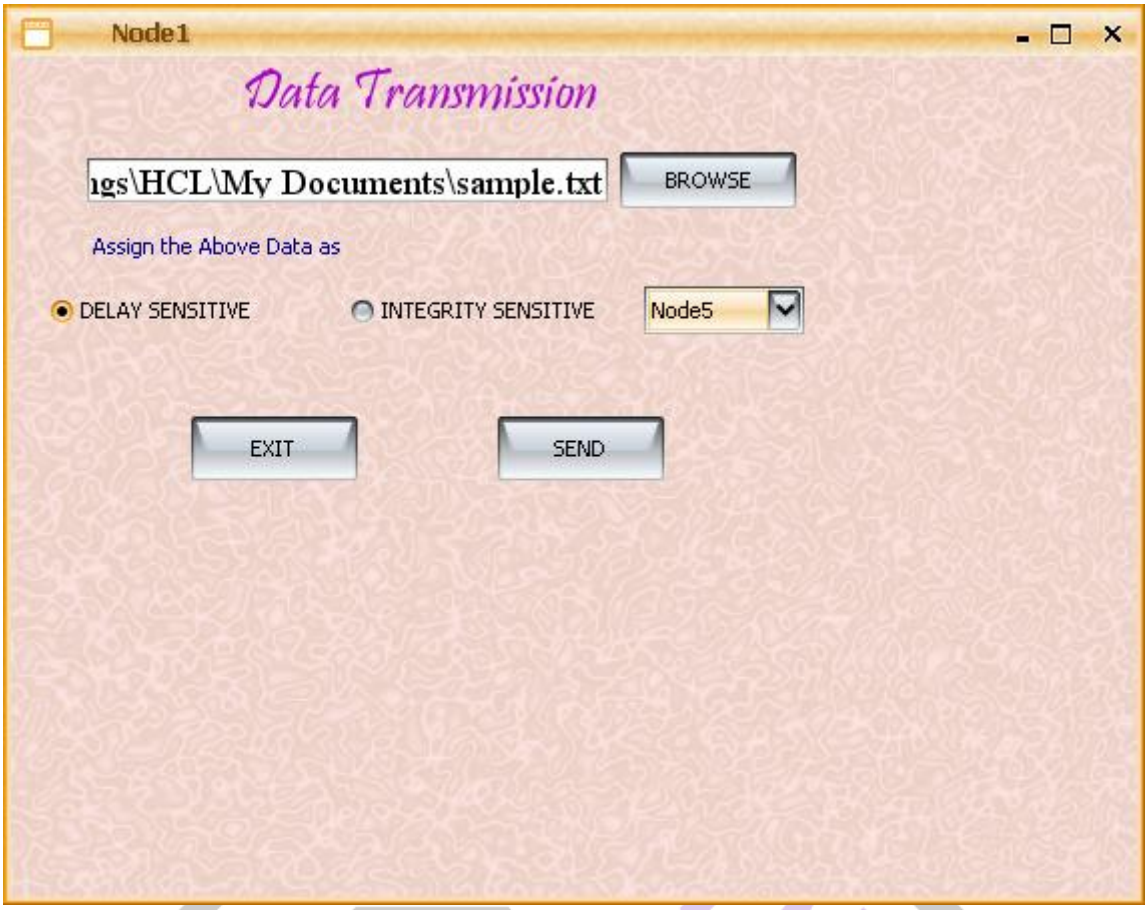

Figure 4 data transmission

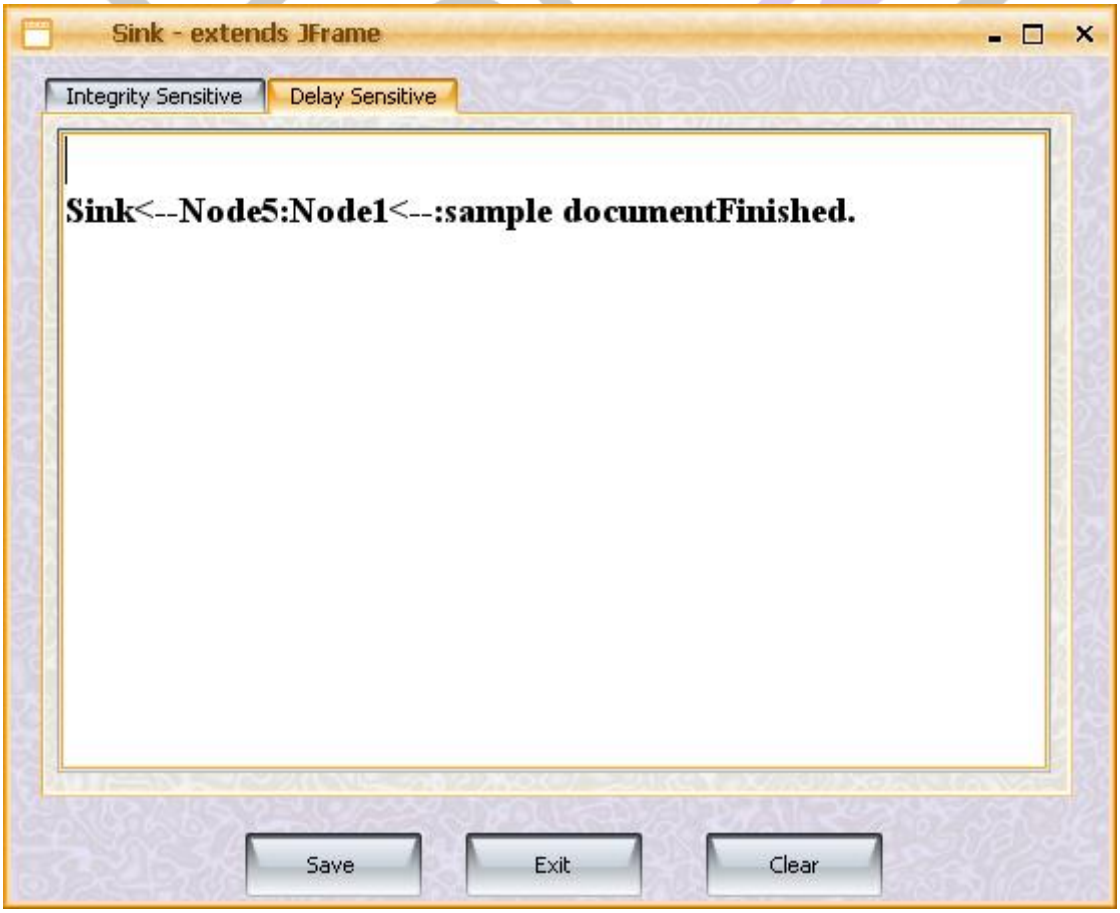

Figure 5 sample Document

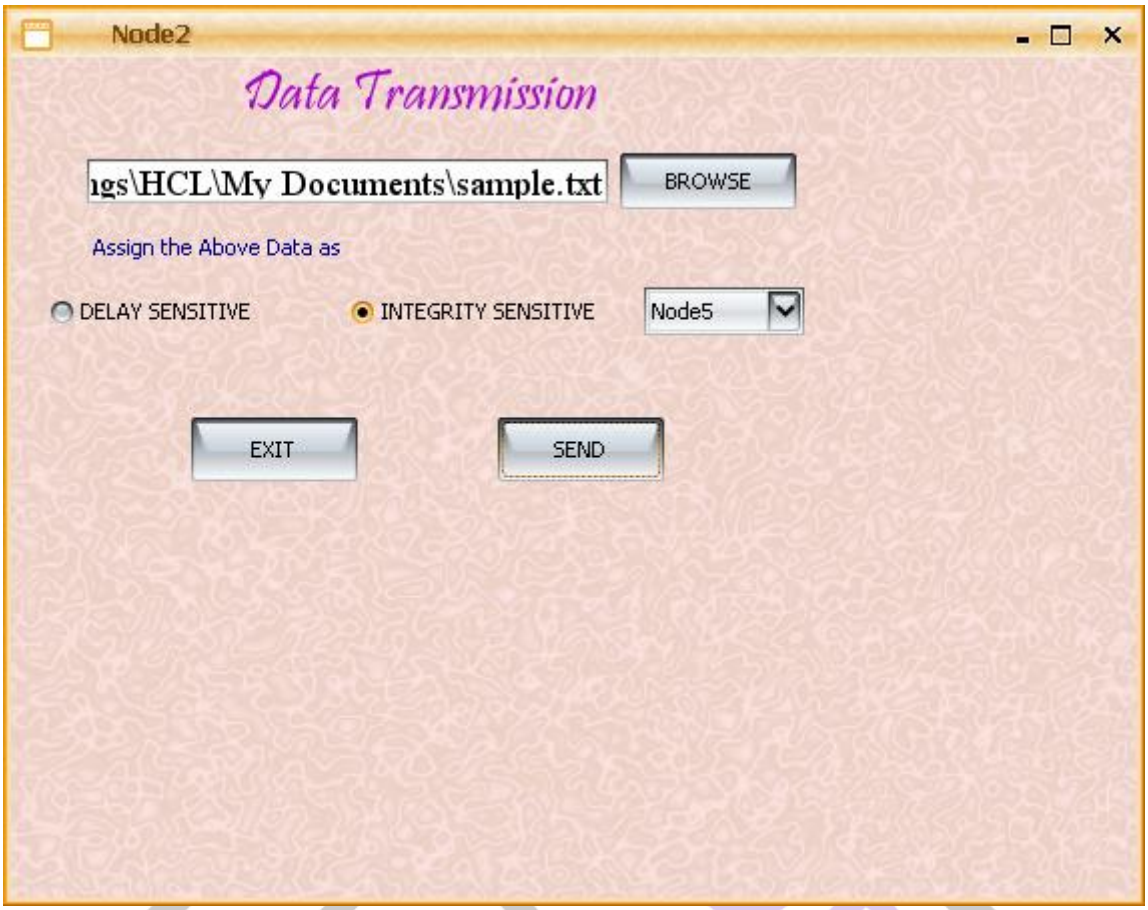

Figure 6 Data Transmission

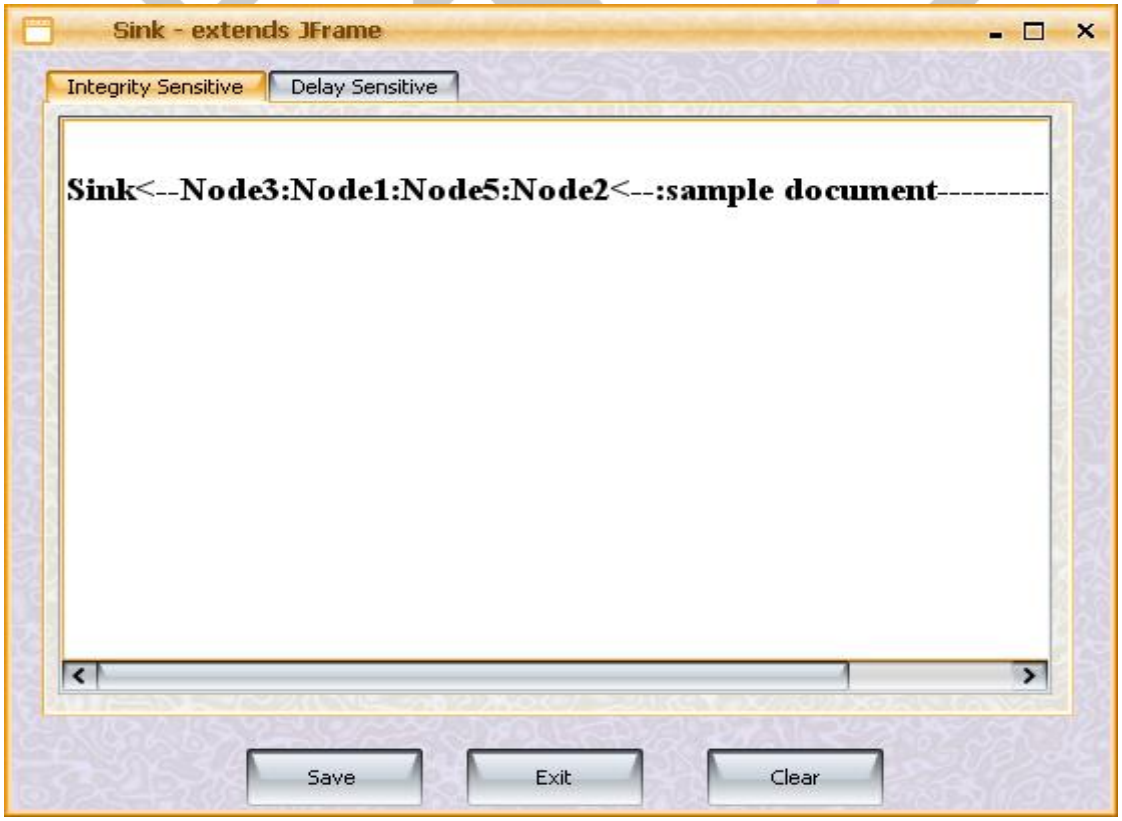

Figure 7 Integrity Sensitive

VI CONCLUSION

A dynamic multipath routing algorithm IDDR is proposed based on the concept of potential in physics to satisfy the two different QoS requirements, high data fidelity and low end-to-end delay, over the same WSN simultaneously. The IDDR algorithm is proved stable using the Lyapunov drift theory. Moreover, the experiment results on a small testbed and the simulation results on TOSSIM demonstrate that IDDR can significantly improve the throughput of the high-integrity applications and decrease the end-to-end delay of delaysensitive applications through scattering different packets from different applications spatially and temporally.
IDDR can also provide good scalability because only local information is required, which simplifies the implementation. In addition, IDDR has acceptable communication overhead.

REFERENCES

[1] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in Proc. 1st Int. Conf. Embedded Networked Sensor Syst., 2003, pp. 126–137.
[2] T. Chen, J. Tsai, and M. Gerla, "QoS routing performance in multihop multimedia wireless networks," in Proc. IEEE Int. Conf. Universal Personal Commun., 1997, pp. 557–561.
[3] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: Core extraction distributed ad hoc routing algorithm," IEEE J. Selected Areas Commun., vol. 17, no. 8, pp. 1454–1465, Aug. 1999.
[4] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," IEEE J. Selected Areas Commun., vol. 17, no. 8, pp. 1488–1505, Aug. 1999.
[5] B. Hughes and V. Cahill, "Achieving real-time guarantees in mobile ad hoc wireless networks," in Proc. IEEE Real-Time Syst. Symp., 2003.
[6] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath multi-speed protocol for QoS guarantee of reliability and timeliness in wireless sensor networks," IEEE Trans. Mobile Comput., vol. 5, no. 6, pp. 738–754, Jun. 2003.
[7] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in Proc. IEEE 8th Real-Time Embedded Technol. Appl. Symp., 2002, pp. 55–66.
[8] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in Proc. IEEE Real-Time Syst. Symp., 2002, pp. 39–48.
[9] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," in Proc. IEEE 23rd Int. Conf. Distrib. Comput. Syst., 2003, pp. 46–55.
[10] P. T. A. Quang and D.-S. Kim, "Enhancing real-time delivery of gradient routing for industrial wireless sensor networks," IEEE Trans. Ind. Inform., vol. 8, no. 1, pp. 61–68, Feb. 2012.