IMPLIMENTATION OF WIRELESS AD-HOC NETWORK USING AURDINO CONTROLLER

P. SRI DURGA RADHA*

D. SUSMITHA RAO*

S. RAJITHA*

Y. SHARMILA*

T. SAILAJA*

K MURALI**

ABSTRACT:

The main aim of the paper is to explain the working of a robot with the use of an Ad-Hoc network. This paper explains major features of the Ad –Hoc network along with the wheel rotational analysis and the DC motor specifications used for the robot. The microcontroller used here is an aurdino controller. This robot is designed in order for access to areas that are not safe for humans in case of disasters. Ad-Hoc is used instead of infrastructure mode because it moves convenient. Secondly we also wanted to test the robot communication using the Ad-hoc network. A brief description of the dc motor and wheel rotational analysis are given with respective our robot construction.

Keyword: Ad-Hoc, capacity, 802.11, traffic density, D C motor, load characteristics, speed control, wheel rotation.

* VIJAYA INSTITUE OF TECHNOLOGY FOR WOMEN, ENIKEPADU, VIJAYAWADA.

** Asst Professor of ECE, VIJAYA INSTITUE OF TECHNOLOGY FOR WOMEN, ENIKEPADU, VIJAYAWADA.

1. INTRODUCTION:

The robot is designed with the use of aurdino controllers interfaced with the Wi-Fi shield for communication. The robot primarily uses Ad-Hoc network since it provides convenient infrastructure free communication. In the first section we discuss Ad-hoc network and its features and the various problems associated with it. Along with the Ad-Hoc network, access methods like 802.11 are also discussed. By analyzing these topics a clear picture of the network and its limitations can be understood. In this paper the mathematical analysis of the throughput available at each node in the network is also discussed. The various factors that lead to the inefficiency of the network are brought forward. On wireless computer networks, ad-hoc mode is a method for wireless devices to directly communicate with each other. Operating in ad-hoc mode allows all wireless devices within range of each other to discover and communicate in peer-to-peer fashion without involving central access points (including those built in to broadband wireless routers). An ad-hoc network tends to feature a small group of devices all in very close proximity to each other.

The robot developed works with the help of D C motor, so the motors mathematical analysis load characteristics, field windings, speed control characteristics are discussed. These characteristics play an important role in the selection of the motor.

Next the wheel rotation of the robot is discussed and its mathematical analysis is performed. The wheel characteristics are important since it would determine the rotational and angular movement of the robot.

2. Ad-Hoc network:

Wireless ,or single-hop networks which are not based on any fixed structure and where communication occurs via nodes , that is data is transmitted in the form of packets from the source node to the destination node is known as an Ad-Hoc network . A mobile ad hoc network is an autonomous collection of users (nodes) that communicate over wireless links .Due to nodal mobility the network topology changes rapidly and unpredictably [1]The capacity of the of the network depends on the network size, traffic patterns and detailed local and radio patterns. The traffic pattern would determine per node scaling capacity. One can assume that the network size is dependent on the total capacity of the network ,since the total capacity of the network grows with the area it covers due to spatial reuse of the spectrum[2].

Ad-Hoc routing requires that nodes cooperate to forward each others packets through the network. This means that the total throughput available at each node would depend upon 2 factors [1]:

1. The channel capacity2. The forwarding load imposed by the distant nodes.

The above conditions could limit the usefulness of the network. It is important to know about the per node capacity since it would determine the network size. The per node scaling capacity can be referred from Gupta and Kumar's analysis [2] , they explain that the total amount of data that can be simultaneously transmitted for one hop increases linearly with the total area of the Ad-Hoc network . If the node density is constant, this would mean that the total one hop capacity is O(n) where n denotes the number of nodes . However as the network grows larger the number of hops between the source and destination may grow larger depending on the communication pattern. The average path length grows with the spatial diameter of the network or equivalent to the square root of the area or $O(\sqrt{n})$. The total end to end capacity is $O(\frac{n}{\sqrt{n}})$ and the end to end throughput available to each node is $O(\frac{1}{\sqrt{n}})$. Gupta and Kumar also demonstrated the existence of a global scheduling scheme achieving $\Omega(\frac{1}{\sqrt{nlogn}})$ for a uniform random network with random traffic pattern. Its not favorable to the users of the network that the throughput at each node approaches zero as the number of nodes increases.

By analyzing the Ad-Hoc routing protocols the observation is that capacity is the limiting factor. Failure of the network usually occurs due to congestion looses which occur due to high volume of routing queries that are caused by the presence of large number of nodes. This ultimately results in the dropped packet, loss of routing information and consequent misrouting of data.

2.1 802.11 Ad-Hoc Protocol:

IEEE 802.11 is the access protocol used in Ad-Hoc networks. To reduce collisions caused by hidden terminals [2] in the network, 802.11 use a four-way RTS/CTS/Data/Ack exchange. In brief, a node that wishes to send a data packet first sends an RTS (request to send) packet to the destination. If the destination believes the network is idle, it responds with CTS (clear to send). The sender then transmits the data packet, and waits for an ACK (acknowledgment) from the receiver. If a node overhears an RTS or CTS, it knows the medium will be busy for some time, and avoids initiating new transmissions or sending any CTS packets 802.11 RTS and CTS packets include the amount of time the medium will be busy for the remainder of the exchange. Each node uses these times to update its "network allocation vector" (NAV). The NAV value indicates the amount of time remaining before the network will become available. Upon successful receipt of an RTS frame *not* addressed to itself, a node updates its NAV to the

maximum of the time carried in the RTS frame and its current NAV value. Upon receiving an RTS addressed to itself, a node returns a CTS frame only if its NAV value is zero, otherwise no CTS is sent. Hence, a sender will see no CTS if its RTS packet has collided with another transmission at the receiver, or if the receiver's NAV indicates that the network is not available. A node times out and re-sends the RTS if it receives no CTS. 802.11 doubles its back off window each time a timeout occurs; it resets the back off to a minimum value after a packet is transmitted successfully or is dropped after reaching maximum retry limit.

802.11 fail to achieve the optimum chain schedule because an 802.11 node's ability to send is affected by the amount of competition it experiences. For example, node 3 in a 7-node chain experiences interference from 5 other nodes, while node 1 is interfered with by three other nodes. This means that node 1 could actually inject more packets into the chain than the subsequent nodes can forward, as detailed in Figure 5. These packets are eventually dropped at nodes 2 and 3. The time node 1 spends sending those extra packets decreases delivered throughput since it prevents transmissions from subsequent nodes. This unfairness was also noted by Nandagopal et al. [2]; their proposed solution, which tries to give each single-hop flow equal capacity allocation, might raise the efficiency of ad hoc chain forwarding configurations. In addition to allocating bandwidth unevenly, 802.11 back off works badly with ad hoc forwarding. Consider the case when node 4 is in the middle of transmitting a data packet to node 5 and node 1 attempt to initiate transmission to 2. Because of two-hop interference, node 1's RTS packet will be corrupted by node 4's transmission and node 2 will not respond with a CTS. Since node 1 does not know about node 4's transmission, it will back off and retry.

3. D C motor Analysis:

DC motors have a great variety of performance characteristics, offered by the possibilities of shunt, series, and compound excitation and in the relatively high degree of adaptability to control. The DC motors used in our robot is a 12 v motor Functional Requirement [3] **:** DC Gear head motor capable of accelerating a 15lb, two-wheel drive robot with wheel diameters of 3.825" at a rate of 3ft/sec/sec. Top speed required will be around 4 feet/sec[3]

Design Parameters**:** Supplied Voltage = 12Volts, Motor size limited to an overall diameter of approximately 2" and an overall length of not more than 4" .

For a DC shunt wound, motor speed is approximately proportional to armature voltage and inversely proportional to field current and this can be illustrated as RPM ~= constant x Ea/If DC Shunt Wound – Steady state[4].

Increased torque is accompanied by a small decrease in speed resulting in decreased counter emf, which allows increased current through the small armature resistance. Typically, shunt wound DC motors are constant speed, having only about a 5% drop in rpm from no load to full load.

3.1 Summary of load characteristics - DC

Motors

As load is added to the motor shaft:

• The shunt motor operates at almost constant speed.

• The series motor operates at speeds which decrease rapidly.

• The compound motor operates with any degree of droop between these extremes, depending on the relative strengths of series and shunt fields.

3.2 Field Windings

Additional field windings are used to compensate for the effects of sparking at the Brushes [4] and to permit good operation under heavy duty conditions. They are of two types:

Commutating windings:

Sparking at the brushes causes excessive local heating of the brushes and commutator,Leading to burning away of the copper and carbon, and possibly more severe effects. Commutating or interpole windings are used, with brushes in the neutral position, to obtain good commutation and minimize sparking at the brushes.

Compensating windings:

Short term heavy overloads, rapidly changing loads or operation with a weak main field can cause excessive cross-magnetizing armature reaction, and the coil voltage may be high enough to break down the air between the adjacent segments to which the coil is connected resulting in flashover or arcing between segments. [1, pp. 244-246] Compensating or pole-face windings are used to mitigate these effects on machines subject to severe duty cycles (e.g. steel mill motors).

3.3 Motor Speed Control

Methods of DC motor speed control are summarized as follows:

Adjustment of the flux (field current) via shunt field rheostat [4]

• Common with shunt and compound wound motors

• Constant power Adjustment of armature circuit resistance

• Common with series motors

• Constant torque Adjustment of armature terminal voltage Ward Leonard System

• MG set: AC motor drives a DC generator;

• Output of the generator feeds the DC motor armature;

• Control is achieved by varying the generator field voltage

• Common in passenger elevator control

• Flashovers that have not progressed into complete armature failure

4. Wheel rotational analysis:

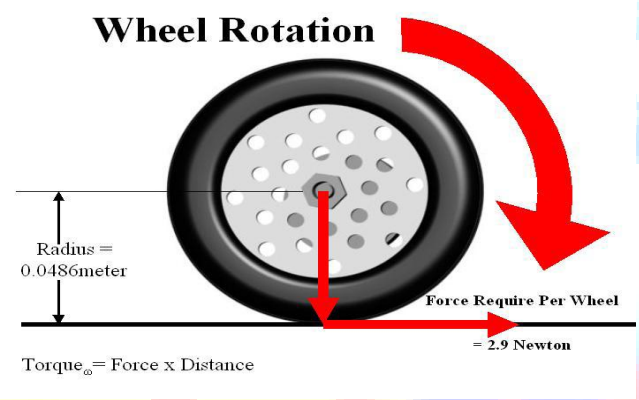

Fig 1: Wheel rotation

A single wheel can only propel itself in the direction of applied force. The simplest possible omni-wheel system that is capable of moving in something other than a straight-line has two wheels attached together so that their directions of driving force are not parallel[4]. Due to the angling of the omni-wheels, by varying the speed at which each wheel rotates, we are able to drive the system in any direction we choose. The velocity of driving wheel one, and $\mathbf{Vin}_{w2}$ is the induced velocity of wheel two. Wheel 1 (right) is driving, while wheel 2 (left) is locked, that is cannot rotate about its axle. Assuming no slippage, only rolling in the direction perpendicular to the driving direction of wheel 2 is possible. In this case, the system will rotate about a single point that must lie along a line perpendicular to the velocity of each wheel. The centre of rotation

may be found at the intersection of the lines perpendicular to V_{w1} and Vin_{w2}. Wheel 2, which provides no driving force, has obtained the velocity Vin_{w2}. This is the induced velocity. where, V_w is the velocity of the wheel, θ is the reference wheel angle, V_{in} is the induced velocity on wheel, φ is the reference body velocity angle, and V_b is the body velocity of robot.

Now V_{in} and V_w are always orthogonal:

$$V_b^2 = V_w^2 + V_{in}^2 \qquad (1)$$

Also:

$$V_{in}^2 = V_b^2 + V_w^2 - 2 V_w V_b \cos(\theta - \varphi)$$

$$= V_b^2 + V_w^2 - 2 V_w V_b (\cos\theta \cos\varphi + \sin\theta \sin\varphi) \qquad (2)$$

Substituting (2) into (1), we may obtain:

$$V_w = V_b(\cos\theta \cos\varphi + \sin\theta \sin\varphi) \qquad (3)$$

For a given rotational velocity of the centre of mass, Ψ&, each wheel must apply velocity:

$$V_w = R\psi . \qquad (4)$$

This is a general equation that is independent of the number of wheels. Consider a three wheeled Omni-directional vehicle with wheels arranged at angles of 0°, 120° and 240°

Wheel 1($\theta = 0$):

$$V_w = V_b \cos\phi + R\psi \qquad (5)$$

Wheel 2 ($\theta = 120$):

$$V_w = V_b\left(-\frac{1}{2}\cos\phi + \frac{\sqrt{3}}{2}\sin\phi\right) + R\psi \qquad (6)$$

Wheel 3 ($\theta = 240$):

$$V_w = V_b\left(-\frac{1}{2}\cos\phi - \frac{\sqrt{3}}{2}\sin\phi\right) + R\psi \qquad (7)$$

5. PROBLEM FORMULATION:

Irregular placement in the Ad-Hoc network leads to some areas of the universe having no nodes. This wastes potential spatial diversity and thus lowers capacity. Random choice of destinations also causes a tendency for more packets to be routed through the center of the network than along the edges. This traffic concentration means that the network as a whole is limited by the capacity of the center [2]. The total one-hop capacity of the network is determined by the amount of spatial reuse possible in the network. Given constant radio range, spatial reuse is proportional to the physical area of the network. Assuming that the node density δ is uniform, the physical area of the network, *A*, is related to the total number of nodes by $A = \frac{n}{\delta}$

Therefore, the total one-hop capacity of the network, *C*, should be proportional to the area, or $C = KA = \frac{Kn}{\delta}$

for some constant *K*. It is approximately 1 Mbps/km^2 for random network simulations.

The traffic pattern in the network has an expected physical path length of *L* from the source to the destination. This means that the minimum number of hops required to deliver a packet is *L/r* where *r* is the fixed radio transmission range. Hence the total onehop capacity in the network required to send and forward packets [2] obeys

$$C > \frac{n\lambda L}{r} nL$$

Therefore, the capacity available to each node, is $\lambda < \frac{Kr}{\delta L} = \frac{C/n}{L/r}$

The above inequality tells us that as the expected path length increases, the bandwidth available for each node to originate packets decreases. Therefore, the traffic pattern has a great impact on scalability. The key factor deciding whether large ad hoc networks are feasible is the locality of traffic.

5.1 DC motor faults

- Poor commutation, sparking at the brushes
- Improperly set kick neutral
- Incorrectly connected fields
- Problems with driven equipment

• Grounded armature circuit
• Grounded field circuit
• Open circuited or cracked commutator risers
• Open circuited armature coil[4]
• Open circuit drive SCR's, Diodes, fuses
• Improper SCR firing circuit operation
• Improperly functioning control circuit
• Control instability

There are also losses due to Copper loss: I2R loss of individual windings and also due to No load rotational losses = Core loss - friction and windage

No load rotational losses: Input power with machine operating at normal speed and excited to produce the calculated internal voltage under full load condition (subtract voltage drops across brushes and series fields).

Friction and windage: Power input at normal speed with machine unexcited

6. RELATED WORK:

This paper is based on a previous paper which is –Capacity Of Ad-Hoc Wireless Networks [2]. The paper on the capacity of the Ad-Hoc network provided lucid data about the features of the network along with its various limitations. It provides clear analysis of the network in random traffic pattern along with the comparison of the efficiency of the nodal mobility in large and small networks.

6.1 Capacity of a Chain of Nodes

In an ad hoc network, packets travel along a chain of intermediate nodes toward the destinations. The successive packets of a single greedy connection interfere with each other as they move down the chain, forcing contention in the MAC protocol. This subsection examines the realizable capacity of a single chain of nodes where packets originate at the first node and are forwarded to the last node in the chain.

The following analysis shows that an ideal MAC protocol could achieve chain utilization as high as 1/3. However, if one assumes that radios can interfere with each other beyond the range at

which they can communicate successfully, the situation is worse. For example, 802.11 nodes in the *ns* simulator can correctly receive packets from 250 meters away, but can interfere at 550 meters. Node 1 is the source of data traffic and the last node in the chain is the traffic sink. Node 1 sends data as fast as its MAC allows. A chain of only two nodes achieves a throughput of about 1.7 Mbps for 1500-byte packets, rather than 2 Mbps, due to the overhead of headers, RTS, CTS, and ACK packets.

802.11 fail to achieve the optimum chain schedule because an 802.11 node's ability to send is affected by the amount of competition it experiences. For example, node 3 in a 7-node chain experiences interference from 5 other nodes, while node 1 is interfered with by three other nodes. This means that node 1 could actually inject more packets into the chain than the subsequent nodes can forward, as detailed in Figure 5. These packets are eventually dropped at nodes 2 and 3. The time node 1 spends sending those extra packets decreases delivered throughput since it prevents transmissions from subsequent nodes. This unfairness was also noted by Nandagopal et al, their proposed solution, which tries to give each single-hop flow equal capacity allocation, might raise the efficiency of ad hoc chain forwarding configurations.

7. PROPOSED METHOD:

The hardware used for the implementation of the Ad-Hoc mode in our project was an Aurdino controller. Arduino is an open-source single-board microcontroller, descendant of the open-source Wiring platform, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board I/O support. The software consists of a standard programming language compiler and the boot loader that runs on the board

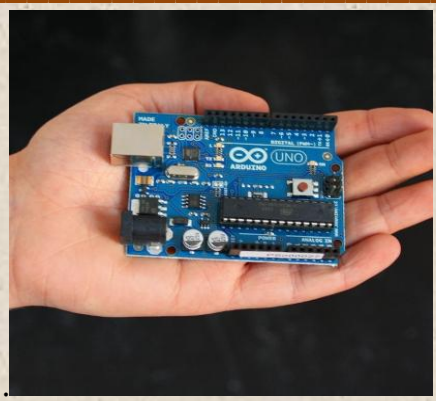

Fig 2: ARDUNIO BOARD

Arduino hardware is programmed using a Wiring-based language (syntax + libraries), similar to C++ with some simplifications and modifications, and a Processing-based IDE.

An Arduino board consists of an 8-bit Atmel AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules (known as shields). Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external chip programmer. At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a simple inverter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.)

7.1 How is Ad-Hoc established?

Here by utilizing the aurdino controller which is interfaced with a Wi-Fi shield we can establish Ad-Hoc mode. The aurdino is placed on the robot and first through RS 232C serial interface communication is established between the laptop and the robot. Through this serial communication a differential program is loaded onto the aurdino controller. This program would control the movement of the robot as per the users command.

The program is written in simple C and C++ and is complied in the aurdino. The aurdino has an in built bootloader that helps in loading the program onto the controller aurdino.exe. The differential program that is loaded is as follows:

PROGRAM:

```
#include "Wi-Fi.h"

byte incoming;

int motorpin1= 9;

int motorpin2= 6;

int motorpin3= 5;

int motorpin4= 3;

void setup()

{
pinMode(motorpin1,OUTPUT);   pinMode(motorpin2,OUTPUT);

pinMode(motorpin3,OUTPUT);

pinMode(motorpin4,OUTPUT);

Serial.begin(9600);

Serial.println("SPI UART on WiFly Shield terminal tool");
```

```
Serial.println("-----------------------------------");

Serial.println();

Serial.println("This is a tool to help you troubleshoot problems with the WiFly shield.");

Serial.println("For consistent results unplug & replug power to your Arduino and WiFly shield.");

Serial.println("(Ensure the serial monitor is not open when you remove power.)");

Serial.println();

Serial.println("Attempting to connect to SPI UART...");

SpiSerial.begin();

Serial.println("Connected to SPI UART.");

Serial.println();

Serial.println(" * Use $$$ (with no line ending) to enter WiFly command mode. (\"CMD\")");

Serial.println(" * Then send each command followed by a carriage return.");

Serial.println();

Serial.println("Waiting for input.");

Serial.println();

}

void loop() {

  // Terminal routine

  // Always display a response uninterrupted by typing

  // but note that this makes the terminal unresponsive
```

```
// while a response is being received.
digitalWrite(3,LOW);
 digitalWrite(5,LOW);
 digitalWrite(6,LOW);
 digitalWrite(9,LOW);
 while(SpiSerial.available() > 0) {
  incoming=SpiSerial.read();
  if(incoming == byte('F')){
  digitalWrite(motorpin1,HIGH);
  digitalWrite(motorpin2,LOW);
  digitalWrite(motorpin3,HIGH);
  digitalWrite(motorpin4,LOW);
  delay(1000);
    }
 if(incoming ==byte('L')){
      digitalWrite(motorpin1,HIGH);
      digitalWrite(motorpin2,LOW);
      digitalWrite(motorpin3,LOW);
      digitalWrite(motorpin4,LOW);
     delay(1000);
   }
```

```
if(incoming ==byte('R')){
digitalWrite(motorpin1,LOW);
digitalWrite(motorpin2,LOW);
digitalWrite(motorpin3,HIGH);
digitalWrite(motorpin4,LOW);
delay(1000);
}
if(incoming == byte('S')){
digitalWrite(motorpin1,LOW);
digitalWrite(motorpin2,LOW);
digitalWrite(motorpin3,LOW);
digitalWrite(motorpin4,LOW);
delay(1000);
Serial.print(incoming,BYTE);
}
if(Serial.available()) { // Outgoing data SpiSerial.print(Serial.read(), BYTE);
}
}
```

After loading the program onto the controller the USB connection between them is terminated. The main purpose of the project is to design a mobile robot without any wired connections therefore the aurdino is interfaced with the Wi-Fi shield. This Wi-Fi shield helps in the establishment of wireless communication between the robot and the laptop. Here the shield

introduces another in built software known as teratermpro. Teratermpro first establishes serial communication by selecting a particular serial port. Next it allows the user to enter the command mode by issuing "$$$" command.

Once this command is accepted by the teratermpro software then it switches the user to command mode. Next the SSID of the user is entered and then a wireless communication between the robot and the laptop via the wi-fly shield is established. By the use of this network the user can issue commands for the movement of the robot.

7. CONCLUSIONS AND FUTURE WORK:

By the use of this network the user can issue commands for the movement of the robot.

The same mobile robot using aurdino can be used in space applications in order to monitor the conditions of the space ships and satellites also same method can be used to establish the internet facility to remote villages.

REFERENCES:

- Ad Hoc Networking Critical Features and Performance Metrics Madhavi W.Subbarao – Wireless Communication Technology Group ,NSIT
- Capacity of Ad Hoc Wireless Networks Jinyang Li Charles Blake Douglas S. J. De Couto Hu Imm Lee Robert Morris .M.I.T. Laboratory for Computer Science jinyang, cblake, decouto, hilee, rtm_@lcs.mit.edu
- Understanding and Using DC Motor Specifications- GEARS Educational Systems 105 Webster St. Hanover Massachusetts 02339 Tel. 781 878 1512 Fax 781 878 6708
- Omni-drive robot motion on curved paths: The fastest path between two points is not a straight-line -Mark Ashmore and Nick Barnes Department of Computer Science and Software Engineering -The University Of Melbourne.