A Project Report on
Multicasting in eCos

Submitted as part of the requirements of course ICS 212 – Introduction to Embedded Systems

Submitted to:
Dr. Rajesh Gupta

Submitted by:
Maulik Oza (SID#32504699)
Shireesh Verma (SID# 88284288)

Department of Information and Computer Science
University of California, Irvine
# TABLE OF CONTENTS

1. INTRODUCTION ........................................................................................................... 3  
   1.1. ORGANIZATION OF THE PAPER ................................................................. 3

2. BACKGROUND ................................................................................................................. 3  
   2.1. eCos – EMBEDDED CONFIGURABLE OPERATING SYSTEM ......................... 4  
      2.1.1. eCos TCP/IP stack: ............................................................................... 4
   2.2. MULTICASTING: .............................................................................................. 5  
      2.2.1. DSR: ..................................................................................................... 5
      2.2.2. DSDV: .................................................................................................. 6
      2.2.3. AODV: .................................................................................................. 6
      2.2.4. ODMRP: .............................................................................................. 6

3. IMPLEMENTATION DETAILS: ..................................................................................... 6  
   3.1. APPROACH AND INITIAL ISSUES: ............................................................. 6
   3.2. eCos MULTICASTING IMPLEMENTATION: ..................................................... 7  
      3.2.1. Multicasting framework: ....................................................................... 7
   3.3. NETWORK SIMULATOR ............................................................................... 8

4. DISCUSSION .................................................................................................................... 9  
   4.1. APPLICATIONS .............................................................................................. 14
   4.2. CONCERNS ...................................................................................................... 15
   4.3. CONCLUSIONS ............................................................................................... 15

5. APPENDIX: .................................................................................................................. 16
Abstract:
With the widespread use of PDA’s for communicating purposes it has become essentially important to address this issue. One of the means of communication is using multicasting framework for sending messages etc. to a group of people. In this report we highlight the approach taken by us for implementing a multicasting framework for a real time operating system eCos and also some simulation results. We also illustrate the analysis done for improving the multicasting in wireless networks by the use of mobility prediction.

1. Introduction
With the world moving towards realizing the dream of ubiquitous computing [1], embedded devices with networking capabilities are the key towards achieving the vision. In today’s computing universe we have devices that are capable of storing large amounts of data and also having networking capabilities, but to achieve total ubiquity and transparency we need to have all devices capable of performing tasks and communicating and coordinating with many other devices of a similar or different nature. Thus an embedded real time operating system forms the core for such devices. For these devices to communicate with each other the operating environment should have basic networking capabilities.

The aim of this paper is to give an insight into the work done in implementing a multicasting framework that would support wireless devices in the real-time operating system eCos, highlight the simulation that was done to demonstrate the framework as well as the analysis of mobility prediction technique.

1.1. Organization of the paper
Section 2 of this paper focuses on building the background that was needed for actually doing the experiment of implementation. In the subsequent section we discuss the approach taken by us for accomplishing the goal and some initial issues encountered. After that we discuss the actual implementation details with regards the framework as well as the simulation tool. In section 4 we discuss the learning, conclusions and some applications of the framework based on our experience developing the project.

2. Background
Since the project required through knowledge of eCos [2] and fundamentals of multicasting, we had to do some groundwork on these two technologies. We had to start off with studying the operating system with more focus on the TCP/IP stack [3] of the operating system, which is a separate component. We had to also look at the various multicasting schemes and decide on a minimal & optimal one that was needed for implementing the framework for multicasting on the operating system selected. In this section we first discuss in brief the eCos RTOS and then go on to discuss its TCP/IP stack. In the next sub-section we discuss various multicasting protocols for wireless ad-hoc networks.
2.1. **eCos – embedded Configurable Operating System**

eCos is an open-source, royalty-free, real-time operating system targeted to embedded applications and supported by the GNU development tools. One of the key aspects of eCos is its configuration system. It allows the programmer to control what functionality and features are included at runtime. Selecting from package modules, the programmer can layer different functionality, such as an Ethernet driver and networking support or a different scheduling algorithm, according to the needs of the application. Unwanted features can be eliminated easily to reduce the resource footprint. The configuration system also enables developers to employ third-party components to extend and enhance the functionality of the operating system. [2]

Hence because of these features we chose eCos to be the ideal RTOS for a project for ubiquitous devices, where resources are the main constraints. eCos provides various packages that can be configured separately based on your needs. eCos provides different components as packages and it can be customized based on your needs. In the next subsection we discuss the eCos TCP/IP stack [3] that was used for this project.

Embedded systems have inherited the programming practices used in larger systems. Network protocols, and TCP/IP in particular, incorporate programming practices used in larger systems. As discussed in the first part of this article, the history of TCP/IP is one of adapting and modifying the original sources written at the University of California at Berkeley to embedded systems. The Berkeley stack is the basis for most of these ports and is the basis of most of the commercial TCP/IP stacks for embedded systems. Of course, real-time and embedded systems face many issues that are unique. A straight port of the Berkeley stack is not the best implementation for the particular needs of an embedded and real-time system.

### 2.1.1. eCos TCP/IP stack:

TCP/IP Networking for eCos [3] provides a complete TCP/IP networking stack, which is derived from the latest stable release of OpenBSD. The networking support is well tested within the eCos environment. The networking package supports only a few Ethernet drivers as of now. The stack since it is based on FreeBSD TCP/IP stack it is robust but currently supports only,

#### 2.1.1.1. IPv4:

Version 4 of the Internet Protocol that is the current version. It supports 32bit addressing and is being used for 30+ years.

#### 2.1.1.2. TCP:

Transmission Control protocol, the standard protocol used for connection-oriented networks. The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks [5].
2.1.1.3. **UDP:**
User Datagram Protocol, the protocol for connection-less networks that is more standard for wireless networks. This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) is used as the underlying protocol [6].

2.1.1.4. **ICMP:**
Internet Control Messaging Protocol, The Internet Protocol (IP) is used for host-to-host data gram service in a system of interconnected networks. The network connecting devices are called Gateways. Occasionally a gateway or destination host will communicate with a source host, for example, to report an error in data gram processing. For such purposes this protocol, the Internet Control Message Protocol (ICMP), is used. ICMP, uses the basic support of IP as if it were a higher level protocol, however, ICMP is actually an integral part of IP, and must be implemented by every IP module [7].

2.1.1.5. **Raw packet interface:**
Raw packets of data can be exchanged through sockets.

Currently, the networking stack for eCos only supports Ethernet based networking. The network drivers use a two-layer design. One layer is hardware independent and contains all the stack specific code. The other layer is platform dependent and communicates with the hardware independent layer via a very simple API. In this way, hardware device drivers can actually be used with other stacks, if that stack can provide the same API.

2.2. **Multicasting:**
Multicasting as defined by IEEE [8] is, “A technique that allows copies of a single packet to be passed to a selected subset of all possible destinations”. Thus in a so-called “networked” world we would want to send messages between devices for control or coordination. Multicasting capabilities if provided in all embedded mobile devices we can have undeterred communication between groups (of course based on the network availability). In this sub-section we discuss some of the ad-hoc wireless protocols that we studied for helping us in deciding the basic minimum multicasting infrastructure that was needed to be implemented for eCos.

2.2.1. **DSR:**
The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR enables the sender of a packet to determine the sequence of nodes through which the packet must be forwarded to reach the intended destination node, and to route that packet along that sequence of hops by including a source route header in the packet. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that is needed to react to changes in the routes currently in use [9].
2.2.2. **DSDV:**
In DSDV, each node maintains a routing table containing the next-hop information for each reachable destination. Each entry has a sequence number, and if a new entry is given, it prefers the one with the greatest sequence number, or if their sequence is the same, it chooses the metric with the lowest value. Each node advertises an increasing even sequence number for itself [10].

2.2.3. **AODV:**
The Ad hoc On-Demand Distance Vector (AODV) routing protocol is intended for use by mobile nodes in an ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network. It uses destination sequence numbers to ensure loop freedom at all times (even in the face of anomalous delivery of routing control messages), avoiding problems (such as 'counting to infinity') associated with classical distance vector protocols [11].

2.2.4. **ODMRP:**
On-Demand Multicast Routing Protocol (ODMRP) is a multicast routing protocol designed for ad hoc networks with mobile hosts. ODMRP is a mesh-based, rather than a conventional tree-based, multicast scheme and uses a forwarding group concept (only a subset of nodes forwards the multicast packets via scoped flooding). It applies on-demand procedures to dynamically build routes and maintain multicast group membership. ODMRP is well suited for ad hoc wireless networks with mobile hosts where bandwidth is limited, topology changes frequently and rapidly, and power is constrained [12].

These above were certain protocols that were studied. Based on the maximal common we decided to provide certain framework that could be used for implementing these protocols.

3. **Implementation Details:**
In this section we discuss the actual project plan and the implementation details of how we went about accomplishing the goal. We first discuss the approach taken by us for the project and some initial issues that were encountered during the process of the experiment and then we go on to discuss the actual details. The details are sub divided based on the tasks, where we first discuss the details regarding the framework for multicasting and then the simulation details as regards to mobility prediction.

3.1. **Approach and Initial Issues:**
As discussed earlier we first had to start off with getting all the relevant background knowledge that was needed for performing the experiment. After the initial study period we had to go about actually gathering the tools that were required, i.e. the eCos operating system that would work on the initial experimentation platform (Windows 2000), the TCP/IP stack and then integrating the TCP/IP package with the minimal configuration of the operating system for proper functionality. After this stage we would then implement
the appropriate multicasting framework that was conceived during the study. After the framework had been implemented, we would select an appropriate simulation tool and simulate the mobility prediction scheme.

After doing the background study we encountered that eCos TCP/IP stack currently has very little to no support for wireless applications, and hence we decided to go ahead with not being protocol specific for implementing the multicasting framework and to just be as generic as we could be. The other issue that bogged us down was that there were few freely available network simulators that could be used for the simulation of the protocol with mobility prediction. After sorting out these issues we started off with the proposed implementation. We discuss this in the next sub-section.

3.2. eCos Multicasting implementation:

In this sub-section we discuss the multicasting framework that was implemented for eCos as well as the Ethernet driver support that was enabled in the current TCP/IP stack. We first discuss the multicasting framework with some implementation details, i.e. the functions that were implemented in order to provide the framework and then the details of how the Ethernet driver support was enabled using the API that was already provided with the TCP/IP stack of eCos.

3.2.1. Multicasting framework:

As discussed earlier we had identified the basic framework that was needed for implementing any multicasting protocol. In this sub-section we discuss the framework and give some implementation details as to how the framework was implemented.

Multicasting as pointed out is a technique of sending out a single packet to more than one host at a time. So the basic functionality needed would be maintaining a list of all the hosts in a particular group. A mechanism that would allow, hosts to leave and join a group when they want. Also if you want groups to communicate at different ports or reuse the same port a functionality that would allow for the reuse of the ports and finally a fixed value for the multicast packets’ time to live. These are the basic functions that need to be provided for any multicasting protocol to be implemented with ease.

The joinGroup function allows users to join a particular multicast group; the parameters that are needed to join a particular group are the group list and the port through which the group is communicating. After sending a joinGroup request with the right parameters the node (host) gets a notification that it has joined the requested group and now it can send and receive messages from that group.

The leaveGroup function allows the users to leave a particular group that it had previously joined. While sending this request the host has to also send the group list as well as the socket at which it was connected to get the socket disconnected. This function after removing the requested node from the list disconnects the host socket and returns a value saying that the node has been removed from the group.
The reusePort function allows the user to actually allow multiple users use a particular port for communication and hence sending and receiving messages. The other major functions of the framework include functions setTTLvalue and setLoopback, which allow for setting the time to live of a particular packet in a node and setting the loopback i.e. a node sending a packet can receive it also respectively.

After implementation of the basic functions to provide the framework the next thing to do was to enable the Ethernet driver support for the TCP/IP stack of eCos. For this we had to actually go through the code to actually understand the functionality of the driver API and also study the other API that was available for implementing the same. The basic functionality that is provided in the Ethernet driver is that of starting the driver, stopping the driver upon completion of the application. The functionality to enable multicasting is provided by two functions – ether_addmulti () and ether_delmulti (), which allow adding the multicasting support and then deleting the support accordingly. These two functions are provided in the TCP/IP stack of eCos and were used to enable the multicasting support in the Ethernet driver. The code for enabling the support is provided below.

```c
    case SIOCADDMULTI:
    case SIOCDELMULTI:
/* Update our multicast list. */
    error = (cmd == SIOCADDMULTI) ?
        ether_addmulti(ifr, &sc->sc_arpcom) :
        ether_delmulti(ifr, &sc->sc_arpcom);

    if (error == ENETRESET) {
        /*
           * Multicast list has changed; set the hardware filter
           * accordingly.
           */
        eth_drv_stop(sc); /* XXX for ds_setmcaf? */
        eth_drv_start(sc);
        error = 0;
    }
    break;
```

The only problem with enabling this is that the implementation becomes platform dependent as eCos currently supports only four Ethernet drivers. The driver works in the promiscuous mode, i.e. it receives all the packets that are within its range and not just the packets that are addressed to it. For coming over this problem there has to be a filter that can be applied, which currently has not been done.

### 3.3. Network Simulator

Now the next step was to search for a simulation tool, which solves our purpose. NS-2 [13] is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. So we decided to use it. It takes a Tcl script which
simulates a topology which is the given to the “nam” animator which then takes the simulation generated and demonstrates it with animation, it shows the transit of packets and the active links.

We then did a simple simulation within the framework we had in NS-2. Currently NS-2 supports Dynamic Source Routing and Destination Sequenced Distance Vector protocols. We used Dynamic Source Routing protocol for our purpose. We wrote a Tcl script, which simulated Dynamic Source Routing Protocol. We took a sample scenario file, which simulates a typical wireless situation. It has a fixed number of hosts and a specified movement pattern for them. Initially we could not get our hosts to multicast but later we got hold of a fix. It is more of a hack rather than a fix, here what we do is that the first hop router before base-station would send the multicast packets encapsulated as unicast to the base-station, thus a base-station does not need to be multicast enabled to receive the packets. Our simulation demonstrated the multicasting between three hosts so that we could visualize and understand the results properly. We tried to generate scenario files ourselves using the scengen generator so that we could test our simulation in different situations but we were unable to generate the scenario files, as the scengen compiler was specific to certain versions of NS-2, which we could not obtain.

4. Discussion

As we have discussed earlier there is a wide choice of multicasting protocols available to us from which we can choose the one, which suits our specific wireless application needs. The choice depends on many factors like, resources available to mobile devices, location and the environment in which it is to be used etc. For instance if a mobile host does not have the luxury of a large storage space then an ideal choice would be Dynamic Source Routing protocol where packet to be transmitted itself contains the routing table with all the information about the hops it has to go through instead of something like Destination Sequenced Distance Vector protocol where each host maintains a routing table.

So for wireless applications the best choice would be the On-Demand Multicasting Routing protocol. As discussed earlier the On-Demand Multicasting Routing protocol delivers data to multicast members using a mesh structure as against a tree-based structure [12, 14]. The source establishes and updates group membership and multicast routes on demand. A query phase and a reply phase construct routes from source to receivers and build a mesh of hosts, which is called a “forwarding group”. On-Demand Multicasting Routing protocol requires periodic flooding to build and refresh routes.

Then our next focus was on mobility prediction. We started with an analysis with the aim of simulating it. There is no implementation available for the scheme so far. But the parameters involved were too complex and cumbersome to deal with.

The idea behind mobility prediction is to be able to predict the network topology at runtime so that routes can be rebuilt and refreshed in case earlier ones are no longer active. The wireless networks entail a periodic flooding of packets, which is not at all desirable because of limited bandwidth available. Apart from that the excessive flooding causes congestion, contention, and collisions. So we need some mechanism to find the
optimal refresh interval, which is not very taxing on the network performance. With mobility prediction we aim at the same goal. The scheme adapts the route refresh interval to mobility patterns and speed. It is also aimed at reducing overheads and transmission of control packets.

However this scheme involves some physical parameters of the mobile hosts to find the time after which two of them go out of the transmission range of each other. It requires the position coordinates; speed and the direction of motion i.e. the angle made by the direction of motion and the base. The parameters are supplied by the GPS (Global Positioning System). We can then predict the duration of time the routes will remain active after a computation. With the predicted time of route disconnection the packets are only flooded when route breaks of ongoing data sessions are inevitable.

In this prediction method a free space propagation model is assumed where the received signal strength is dependent on its distance to the transmitter. In other words, the attenuation of the signal is inversely proportional to the square of the distance between the receiver and the transmitter. Other important assumption is that all the hosts in the network have their clocks synchronized. So if we have dynamics of two neighboring hosts (such as speed, direction, radio propagation range etc.), we can obtain the span of time these two nodes will remain connected.

In order to obtain the expression for the time they will stay connected, let us do a computation. Let the two mobile hosts be ‘i’ and ‘j’ and assume the following about them.

- The transmission range for each of them is $r$
- Their coordinates are $(x_i, y_i)$ and $(x_j, y_j)$ respectively
- Their speeds are $v_i$ and $v_j$ respectively
- Their directions of motion are $\Theta_i$ and $\Theta_j$, respectively such that $0 \leq \Theta_i, \Theta_j < 2\pi$

$T$ is the amount of time they stay within their ranges and after which they go out of range.
This schematic is shown in the figure. As regards the figure shown consider,

x-component of their relative velocity, \( a = v_i \cos \Theta_i - v_j \cos \Theta_j \)
y-component of their relative velocity, \( c = v_i \sin \Theta_i - v_j \sin \Theta_j \)
Distance between them in the x-direction, \( b = x_i - x_j \)
Distance between them in the y-direction, \( d = y_i - y_j \)

So at any time the distance between the hosts is \( D = \sqrt{b^2 + d^2} \).
For the link between them to be active we must have \( D < r \).

So with our assumptions at time \( T \),

\[
(b + a T)^2 + (d + c T)^2 \leq r^2
\]

If we solve this, it gives a quadratic expression for \( T \).

\[
T = -\frac{(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}}{a^2 + c^2}
\]

It has a huge computational overhead. Its complexity is of the order of \( O(N^2) \), which is demanding on the resource deprived mobile hosts. There are other considerations involved.

Although in general the routes are selected on basis of minimum delay criteria but for this schematic we use the most stable route criteria. The most stable route is the one whose expiration time is the latest. The host scans all the possible routes and after an estimation of the time after which the routes expire, it chooses the route with latest expiration time. This scheme minimizes the disturbances caused by mobility since a different route with a greater expiration time is used before the earlier route gets disconnected. This scheme although is efficient but it increases the route refresh time and hence the latency of the network increases.

Apart from this, the GPS can provide the coordinate information but obtaining the velocities seems to be tough in the real scenario although some value might be obtained from the device. This means we proceed with the assumption that hosts have simple mobility patterns and there are no sudden changes of direction and constant velocity. This assumption however, cannot hold in majority of scenarios in real life. A host may have acceleration, deceleration and change direction during its motion. Also we assume a linear two-dimensional motion, which is a drastic simplification if we are to consider real life scenario in strict sense.

So this scheme may not be a very pragmatic solution to the adhoc networks. However if we relax the real life constraints and make some simplifying assumptions it leads to noticeable improvements. So below we present a sample scenario where mobility prediction can indeed be simplified and be applied.
**Application scenario when Mobility prediction can be solved:**

A typical scenario where the mobility prediction scenario could be solved is, PDA’s communicating through some central base station within a certain area, the PDA’s send data to the base station and the base station then multicasts it to the hosts that the PDA has asked to be sent to. This would then not actually require the base station to be running eCos and hence would not require the multicast framework that was developed but instead can use the standard libraries that are already available for multicasting in more scalable operating systems.

As described above the mobility prediction equation for two mobile devices with coordinates \((x_i, y_i)\) & \((x_j, y_j)\), and velocities \(v_i\) & \(v_j\), moving at \(\Theta_i\) & \(\Theta_j\) angles respectively takes the following form:

\[
T = -(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}
\]

where,

\[
\begin{align*}
    a &= v_i \cos \Theta_i - v_j \cos \Theta_j \\
    b &= x_i - x_j \\
    c &= v_i \sin \Theta_i - v_j \sin \Theta_j \\
    d &= y_i - y_j
\end{align*}
\]

Now since we have one node always stable, i.e., non-mobile the velocity and the angle components are 0 and a fixed coordinate, suppose we assume that to be the origin of the universe i.e. \((x,y)\) of the base station to be \((0,0)\). So, the equation now simplifies to,

\[
T = -(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}
\]

where,

\[
\begin{align*}
    a &= v_i \cos \Theta_i \\
    b &= x_i \\
    c &= v_i \sin \Theta_i \\
    d &= y_i
\end{align*}
\]

Now, we go on to solve it for a sample scenario. Consider, a sample scenario where we have the base station maintaining information about the PDA’s that are connected to it.

Range of transmission of the base station assumed is: 500 meters. The coordinates that are specified are in meters and are with respect to the base station assuming the base station to form the origin of the coordinate system. The velocities are specified in meters/second, now consider the following table with the values, the direction of motion is with respect to the positive x-axis. The signs in the velocity components have been ignored because it has been taken care off in the expression for mobility prediction.

<table>
<thead>
<tr>
<th></th>
<th>Coordinates (b,d)</th>
<th>Distance from Origin</th>
<th>Velocity (Direction)</th>
<th>Vx (a)</th>
<th>Vy (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDA1</td>
<td>(20,100)</td>
<td>~102</td>
<td>5 (30)</td>
<td>4.33</td>
<td>2.5</td>
</tr>
</tbody>
</table>
The scenario can be visualized as shown in figure:

```
<table>
<thead>
<tr>
<th>PDA</th>
<th>x, y</th>
<th>d</th>
<th>v</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDA1</td>
<td>(10, 50)</td>
<td>~51</td>
<td>6 (45)</td>
<td>4.24</td>
<td>4.24</td>
</tr>
<tr>
<td>PDA2</td>
<td>(-10, 90)</td>
<td>~90.5</td>
<td>3 (120)</td>
<td>1.5</td>
<td>2.59</td>
</tr>
<tr>
<td>PDA3</td>
<td>(-40, -60)</td>
<td>~72</td>
<td>9 (235)</td>
<td>5.16</td>
<td>7.37</td>
</tr>
<tr>
<td>PDA4</td>
<td>(50, -30)</td>
<td>~58</td>
<td>8 (330)</td>
<td>6.92</td>
<td>4</td>
</tr>
</tbody>
</table>
```

The base station maintains the information and the speed & location information about the PDA’s is transmitted by the PDA’s using GPS technology. Now calculating the estimated times for which the link will be active can be given by,

For PDA1:

\[
T = \frac{-(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}}{a^2 + c^2}
\]

where,

- \(a = 4.33\)
- \(b = 20\)
- \(c = 2.5\)
- \(d = 100\)

\(T \sim 85\) secs

Similarly PDA2:
Similarly PDA3:
\[ T \sim 76 \text{ secs} \]

Similarly PDA4:
\[ T \sim 138.5 \]

Similarly PDA5:
\[ T \sim 50 \text{ secs} \]

Thus, the base station need not worry about the links to be refreshed in the table for the mentioned amount of time by ‘T’ for the specific PDA. The base station based on these calculated values needs to predict the approximate times when the links to these nodes should be refreshed. Thus the above is a sample scenario where the mobility prediction scheme does work but then there are no results to actually evaluate the mechanism.

Considering some of the limitations discussed earlier and the fact that GPS may not work properly in certain situations (for instance, indoor, fading), we may not always be able to predict the expiration time of a particular link. [15] A mobile host is made to measure transmission power samples from the packets arriving to it from its neighbor. The host knows in advance the exact strength of the power signal emanating from the source. With this information it computes the rate of decay of signal power for a particular neighbor. Therefore it can predict the time when the transmission power level will drop below the threshold level required for transmission. So this scheme can provide a metric of the time after which the routes should be refreshed.

We surveyed two simulations using two protocols as bases; they are Distance Vector Routing and Dynamic Source Routing. In case of Distance Vector Routing, [16] the most recent routing information is maintained by exchanging route tables with the neighboring hosts. The performance in this case is very sensitive to periodic update interval. In this case the triggered update transmissions are completely gotten rid of. So the routing update interval is relaxed and the updates are required less frequently. In order to propagate the mobility information, a mobility vector field is added to the route update packet. Also the route expiration time is incorporated into the routing table entries. While in case of Dynamic Source Routing protocol data is delivered to multicast members using route whose information is provided in the data packets themselves.

### 4.1. Applications

Now, since we have implemented the multicasting framework for eCos TCP/IP stack so the question is what is the use of multicasting. The framework we have developed is easy to fit in and to be used in a plethora of applications. Some of these applications can be,

- **Military environments** – Here the possible use can be for the communication between the ground troops and also for the air forces.
- **Emergency operations** – In this case it can be use for search and rescue operations where the workers need to work in coherence with each other so they are well informed of each others progress and they know which direction to further proceed in.
Civilian environments – In this scenario the possible use could be in places like meeting rooms, stadiums or university campuses where the workers need to talk to each other for expedition of their work.

4.2. Concerns

However there are concerns associated with the ad-hoc networks which limit the scope of the applications to which multicasting framework can be put to. Some of the important ones are as follows.

- High degree of mobility – The dynamic nature of such networks is one of the foremost hurdles.
- Intermittent connections – Since there is no dedicated connection like Ethernet available, the connectivity is not always guaranteed.
- Bandwidth limitations – Since it is a wireless network, it uses radio frequency and there are restrictions over the bandwidth available.
- Limited power – Also the small mobile devices have tough time managing their power. They have limited battery supply for all their computational needs from which they have to spare some amount for their network-based computations.
- Limited memory – The small mobile devices have a very small amount of memory and storage available to them, in which all the computation has to fit in.

If we use a complex algorithm for routing and multicasting which requires sizeable amount of computation on the part of the mobile device for it to maintain connectivity then it leaves a little memory with the device for its other computing needs.

4.3. Conclusions

After delving so much into eCos we can very safely say that the TCP/IP stack of eCos needs lots of improvements. Especially if we consider adhoc wireless networks it requires many enhancements. Multicasting we implemented is pretty shaky and highly dependent on the target (Ethernet driver support). Apart from that we could get it working only for the promiscuous mode.

The performance of the whole TCP/IP stack in eCos is again highly target dependent and varies with different Ethernet controllers. At present only four controllers are supported on eCos platform. They are Motorola PowerPC MBX/860, Cirrus Logic EDB72xx, based on the Crystal CS8900, Socket Communications Low Power Compact Flash Ethernet adaptor, Intel EBSA-285 + EtherPRO 10/100+.

Mobility prediction as we have already discussed is good for ad-hoc wireless networks in certain scenarios. It might not be always able to predict the network topology. One other limitation of the scheme is that it is a resource-consuming operation to calculate and hence might not suitable for mobile devices that are actually resource deprived.
5. APPENDIX:

MACRO SUPPORT FOR ECOS

Authors:        Maulik & Shireesh
Description:    This provides the basic framework for Multicasting in eCos. It provides the basic functions that need to be implemented by all the multicasting protocols like joinGroup, leaveGroup, reusePort, setTTLvalue etc.
Copyright:      This is the copyright of the authors and use of this source is granted provided the full header is included.

#ifndef MULTICAST_H
#define MULTICAST_H

/* include files standard files that are provided with eCos C language support*/
#include <stdio.h>
#include <string.h>

/* The types, socket and other files for networking support from the networking package*/
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioct1.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>

/* define statements */
#define MULTICAST_ADDRESS_BASE  "225.0.0."
#define MULTICAST_PORT_BASE    30000
#define DEFAULT_MULTICAST_TTL_VALUE  32 /* to send datagrams to any host anywhere */

#ifndef INADDR_NONE
#define INADDR_NONE 0xffffffff
#endif /* INADDR_NONE */

#define NAME_LEN 100
#define MAX_LEN 1024
#define TRUE 1
#define FALSE 0

#endif /* MULTICAST_H */
/* functions to support the essential multicasting functions */
extern void leaveGroup(int recvSock, char *group);
extern void joinGroup(int s, char *group);
extern void reusePort(int sock);
extern void setTTLvalue(int s,u_char *i);
extern void setLoopback(int s,u_char loop);

/* external variables */
extern int   errno;
extern char  *sys_errlist[];

#endif /* MULTICAST_H */
References:


