Cheapest Paths in Multi-Interface Networks
A Distributed Approach

Harish Muthuveeran Shanmugam
Department of Engineering and Information Sciences
Middlesex University, Greater London
Greater London, United Kingdom
msdharish@yahoo.co.uk

Anjana Prasad
Department of Engineering and Information Sciences
Middlesex University, Greater London
Greater London, United Kingdom
anjanaprsd@gmail.com

Abstract

Let G = (V,E) be a graph which models a set of wireless devices (nodes V) that can communicate by means of multiple radio interfaces , according to activating the common interface rule at each node. Every Interface can be modeled as the (Edges E). A connection is satisfied (activated) between any two nodes in the network if each pair of node at least has a single wireless interface in common. In general, every node holds a subset of all possible *k* interfaces. Such networks are known as multi-interface networks. In such networks, we study and analyze the basic problem called *Cheapest Paths.* Shortest path problem is a well known concept in graph-theory. Standard shortest or the cheapest path algorithm such as the *Distributed Bellman-Ford Algorithm* were used to determine the cheapest path in the multi-interface scenario.

K*eywords:* Distributed Algorithm, Energy Saving, Multi-Interface network

I. Introduction

With the advancement in the field of communication engineering more devices with attractive built-in functions have been developed to deal with communication efficiency. Nowadays Wireless devices such as laptops, mobile phones etc. are equipped with multiple radio interfaces to handle variety of connections, for example Bluetooth for Personal Area Networks, Wi-Fi for Local Area access and GPRS for wide area access and communication. The selection of *"best"* radio-interface for a specific connection might depend on a variety of factors. Namely, availability of specific radio interfaces in each device, required communication bandwidth, the *cost* (in terms of energy consumption) for activation and maintaining a common interface, the available neighbors and so forth. Each wireless interface such as Bluetooth, Wi-Fi, GPRS, Infrared, UMTS and GPS have different activation costs in terms of energy consumption. While managing such connections, a lot of efforts are dedicated to energy consumption issues. Due to this fact optimization problems are very challenging in wireless networks. Generally speaking, given a graph G = (V,E), where V represents the set of wireless devices, each $v \epsilon$ V is associated with a set of available interfaces I(v) and E is the set of possible communication links according to the proximity of the available neighboring nodes in the network. The problem in general can be stated as follows. Given a parent node *s* and target node *t* in G, what is the most economical way, i.e., which of the available common interfaces to be activated in some nodes in-order to guarantee a path between node *s* and *t*. Note that a connection is satisfied between any two nodes in the network if they share at least an activated interface in common. Moreover, for each $v \epsilon$ V, there is a set of available interfaces. From now on it is denoted as I (v). The Universal set of all available interfaces in the network will be denoted as $U_{v \epsilon V}$ I (v). Our main goal is to activate the available minimum set of interfaces at each node starting from the parent node *s*, covering other intermediate nodes and thereby reaching the target node *t* with lesser cost of activation. In this paper we

determine the cheapest path in multi-interface networks using the Distributed version of Bellman-ford algorithm [1, 2].

II. Related Work:

The concept of multi-interface networks has recently been studied under many different contexts, but two contexts have impressed the researchers a lot, they are as follows:

a) Connectivity
b) Cheapest Path

In context to connectivity issue the research engineers have formulated a centralized minimum-spanning tree algorithm for finding solution to the coverage problem of a network modeled as a graph. They have found many interesting and approximation results for different types of graphs such as the complete graphs, graphs of bounded degree and planar graphs based on the issue of connectivity. The *Cheapest Path* problem is a well known problem in the field of operational research. In this context polynomial algorithm based on the famous *Dijkstra Algorithm* were used to determine the cheapest path in the multi-interface network scenario. Many unexplored results can be obtained if the same problems of the cheapest path are extended to a distributed setting based on standard *Bellman-ford Algorithm*. Results related with the standard Dijkstra's algorithm have been obtained for a slightly different case where k is not known in advance, i.e., k depends on real time [2].

III. Definitions and Notation:

This Section provides the definitions and notation to study the shortest path problem in multi-interface networks. [2, 5]

The input is a graph G = (V, E), where $|V|$ = number of nodes in the network and $|E|$ = number communication links or interfaces that are available in each node. The set of neighbors of a node $v \in V$ is denoted by N (u), the set of available interfaces in v by I (v). Let k be the cardinality of the available and distinct interfaces in the network. For each node $v \in V$, we define an appropriate interface assignment function W that guarantees that each network connection (i.e., each edge of G) is covered by at least one interface.

Definition 1: A function $W: V \rightarrow 2^{\{1 \ldots k\}}$, with $k = \left| \bigcup_{v \in V} I(v) \right|$, is said to cover a graph G = (V,E) if for each $(u, v) \in E$ the set $W(u) \cap W(v) \neq 0$.

The cost of activating an interface for a node is assumed to be identical for all nodes and is denoted by the *cost function* c: {1 …k} → , the cost of activating an interface is denoted by c_i.

A path P in G from the parent node s to the target node v is denoted by a sequence of couples: the interface i_j used to reach the node v_j is given. For example the sequence of couples can be, the sequence: $P = < (s \equiv v_0, 0), (v_1 i_1)$,.. $(v \equiv v_t, i_t) >$, denotes a path P from s to v that moves on the nodes $s, v_1, .. v_{t-1}, v$ and that reaches node v_j *by interface* i_j . Interface 0 is used denote "no interface" since the source node cannot be reached by any other node in P. Conventionally, $c_0 = +\infty$. However the source node needs to activate interface i_1 to reach the node v_1 , hence the cost of activating the edge (s, v_1) is $2c_1$. In general the cost for activating the interface in path P is

$$d_p(v) = \sum_{j=1}^{t} \cos t((v_{j-1}, i_{j-1})(v_j, i_j)) \quad \text{equation (1)}$$

Where

$$\cos t((v_{j-1}, i_{j-1})(v_j, i_j)) = \begin{cases} 2C_i \text{ if } (i_{j-1} \neq i_j) \\ C_i \text{ else} \end{cases} \quad \text{equation (1.1)}$$

If the current node v_j and previous nodes v_{j-1} communicate using different interfaces so that $i_{j-1} \neq i_j$, then the cost of activating the interfaces is $2C_i$. In other words, if there is a need for switching between multiple interfaces within a single wireless device, the cost of activating the interfaces is $2C_i$ and the cost C_i remains unchanged if there is no need of switching between different interfaces within the same wireless device. There can be several paths from the starting node's' to the destination node't'. The cheapest path will be the path with minimum cost denoted as $\delta(v)$. As $\delta(v)$ is the cheapest among all the paths from source to destination, we can say that $\delta(v) = min\{ d_p(v) \}$. The cheapest path to a particular node via an interface 'i' can be expressed as $\delta(v,i)$. The algorithm for finding the cheapest path by activating the cheapest interface can be formulated using the equation (1.1).

IV. Assumptions for our Proposed Distributed Algorithm:

There are some assumptions that we had made before going into the details of our proposed distributed cheapest path algorithm, they are as follows:

a) Each node has information only about its directly connected neighbors.

b) Each node transmits and receives message only to its directly connected node.

c) Each node must have at least one common interface to its directly connected neighboring node.

d) If there is any change in the network topology or cost of any particular node, the nodes can broadcast to their neighboring nodes about the change in the network.

e) Each node is capable of sending and receiving the following messages to its neighboring nodes.

1) Explore message – this message is used by the nodes to search the neighboring nodes. In wireless networks the nodes are continuously varying, each node has to continuously search for its immediate neighbors. When a node enters a network it will search for its direct neighbors by sending an explore message to all the nodes in its range. 'Exp_v' is used to denote explore message in the proposed algorithm. The format of explore message sent by the source node is [6].

$$Exp\_v = \{ u, i, n_u \epsilon N \}$$

Where u represents the node which is sending the explore message currently and $n_u \epsilon N$ represents the set of directly connected neighboring node of 'u'.

2) Add message – this message is sent by the nodes who received the Explore message from its neighboring node. After receiving the explore message the node will acknowledge its existence and also provides information regarding the common interface between the connecting nodes. 'Add_u' is used to represent the add message sent by the neighboring nodes. The format add message in the proposed algorithm is given as

$$Add\_u = \{v, \pi (v, i), i, c_i \}$$

Where '*v*' denotes the neighboring node that is sending the add message, $\pi (v, i)$ represents the parent node of '*v*' connected via interface i and the cost of activating the interface is given by 'c_i'.

3) Broadcast message – this message is used by the nodes to broadcast its current status. Any node after getting the estimated cost for activating the interface can broadcast the minimum cost or the current minimum cost of reaching the node. 'BR_v' is used to represent Broadcast message send from a newly added node to its neighboring node. The format of broadcast message is given as

$$BR\_v = \{\pi (v, i) , d (v, i) \}$$

Where $\pi (v, i)$ represents the parent node of v, $d (v, i)$ denotes the estimated cost of activating the interface i at node '*v*'.

V. Our Proposed Distributed Cheapest Path Algorithm:

The proposed distributed algorithm is divided into three sections, Initialization process, Neighbor setup process and cheapest path process. The first part of the algorithm is the general initialization used in Bellman – Ford algorithm.

Algorithm 1. The Initialization Procedure

Process Initialize (Graph G: (V, E) , node : s)

1: d(s) = 0
2: **for each** i ϵ I (*v*) do
3: d(s,i) = 0
4: **end for**
5: **for each** *v* ϵ V, v ≠ s do
6: **for each** i ϵ I (v) do
7: d(*v*,i) = +∞
8: d(*v*) = + ∞
9: **end for**
10**: end for**
11: **Neighbor process { G, s , n ϵ N , │V│- 1 }**

The source node is assigned zero, assuming that the length of any path to source is zero. Next all the interfaces associated with the source node is also given zero, *d (s,i) = 0*. Then the algorithm checks for all other nodes and the interfaces associated with those nodes in the network and assigns infinity to it. In the end, the algorithm calls for the Neighboring setup. [1]

Algorithm 2: Neighboring_ Process

Neighbor_Process {G, set of nodes: u ϵ U, n ϵ N, int : count = │V│, k }

1: Temp =
2: **for each** u ϵ U do
3: **Exp_u (v)** = {u, n_u ϵ N, C_i}
4: Send Exp_u (*v*);
5: Temp = Temp U N (u)
6: **for each** *v* ϵ N (u) do
7: **for each** i ϵ │W (u) ∩ W (*v*) │
8: **Add_v (u)** = {v, π (*v*,i) , (i, C_i) }
9: Send Add_v (u)
10: if d(*v*, i) > d(u, i) + C_i **then**
11: d(*v*,i) = d(u ,i) + C_i
12: π(*v*,i) = u;
13: **end if**
If d(*v*,i) > d(u, i) + 2C_i **then**
14: d(*v*,i) = d(u,i) + 2C_i
15: π(*v*,i) = u;
16: **end if**
17: if d(*v*) > d(*v*, i) **then**
18: d(*v*) = d(*v*,i)
19: **end if**
20: **end for**
21: **if** │W (*v*) ∩ W(v_{k+1}) │
22: **for each** N(v_{k+1}) ϵ *v* do

23: **Br_v** = {$\pi(v,i)$, $d(v,i)$}
24: **Send Br_v ;**
25: **end if**
26: **end for**
27: **end for**
28: **if** - - count ≤ $\left|V\right|$ -1then
29: Neighbor_ Process (G, temp, count)
30: **end if**

In Neighbor process algorithm, the network graph G, source 's', its neighbors and the number of nodes in the graph G are introduced. A 'Temp' variable is assigned as a null set to store the details of the neighboring nodes of a particular node. In line 2 – 4, the algorithm searches for all directly connected neighbors of source node by sending an explore message. For each neighbors of the node the algorithm searches for a common interface. If there is a common interface between those two nodes, then the neighboring node will sent an add message in response to the explore message. This process is then accompanied by the Edge relaxation process in Line 10-20.

In the relaxation process, the algorithm checks for the cheapest activation cost for a single node. If the new cost is lesser than the current cost, then the current cost is replaced with the new cost. If the cost is greater than the current cost then there is no change to the cost and the node remains unchanged. The algorithm keeps on checking until the node gets the cheapest cost.

In the end of relaxation process the current neighboring node will update the routing table of the neighbors of the neighboring node. This is done by sending broadcast messages to the immediate successor nodes. Each node in the network does all the above mentioned process iteratively for (|V|-1) times. The repetition of the loop for (|V|-1) times allows the distances to propagate throughout the network and thus the routing table of every node in the network is updated.

Algorithm 3: Cheapest Path Process

Cheapest_ Path (G, nodes; s, *v* ϵ V; interface: i, j)
1: **for all** *v* ϵ V do
2: **end Exp_v;**
3: **end Add_v;**
4: **end Br_v;**
5: **if** (s ≠ t) then
6: i = 0; d(s, i) = 0;
7: **while** d(*v*) ≠ d(*v*, i)) do
8: i++
9: **end while**
10: **if** $\left|W(u) \cap W(v)\right| \neq 0$; and $d(u, j) + C_j < d(v,i) + C_i$
11: i = j ;
12: **end if**
13: Cheapest_Path (G, s , $\pi(v, i)$, i)
14: return (*v*,i)
15: **else**
16: return (s, 0)
17: **end if**
18: **end for**

The third part of the algorithm is used to find the cheapest path from the source to destination. The cheapest cost of each node and the associated interfaces is found using the Neighbor setup algorithm. The cheapest path algorithm compares this obtained cost and finds the most suitable and cheapest way to reach the destination. For this the algorithm checks if the interface used in the previous node is equal to the interface in the current node. This operation is done by the 'while' loop in the algorithm. It will check whether the interface used to reach the parent node (v') and the interface used to reach the current node (v) are same. If it is same then that path will be considered as a cost effective path and is added to the cheapest path. This process is continued until all the nodes in the cheapest path and the interface used to reach them are covered.

VI. Proposed Algorithm Analysis and Results:

We had analyzed our proposed distributed algorithm using a simple network modeled as a graph. The Network model that we had considered is a simple undirected graph as shown in the Figure.1below.

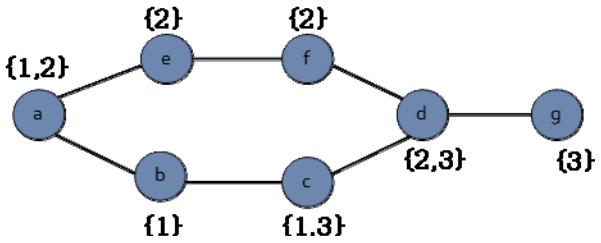

Figure: 1- Network Model

The above network consists of nodes a, b, c, d, e, f, and g and interfaces 1, 2, and 3. The cost of activating interfaces 1, 2, 3 is given as 1.5, 1.5, 1 respectively. Let as consider node 'a' as the source node and node 'g' as the destination node. By using the proposed algorithm we should find the cheapest path from node 'a' to node 'g'.

From the graph, we can see there are two paths from 'a' to 'g'. The upward path a – e – f – d – g and the downward path a – b – c – d – g. We have to find the cheapest among these two paths. First we will find the cost of each path by activating the interface along this path. We can see that in the upward path a – e – f – d – g, interface 2 is used to communicate from 'a' to 'd' and interface 3 is used from 'd' to 'g'. So the total cost of activating the interface along upward path is equal to 1.5 + 1.5 + 1.5 + 1.5 + 1 + 1 = 8.

Taking a – b – c – d – g, the downward path into consideration, the interface used from a – c is interface 1 and c – g is interface 2. So the total cost of activating the interface along downward path is equal to 1.5 + 1.5 + 1.5 + 1 + 1 + 1 = 7.5. Thus we can say that the cheapest path obtained by taking the interfaces into consideration is CP_d = { (a, 0), (b, 1), (c, 1), (d, 3), (g, 3) }. The step by step process of the implementation of the proposed algorithm in the graph is shown below.

Step1:

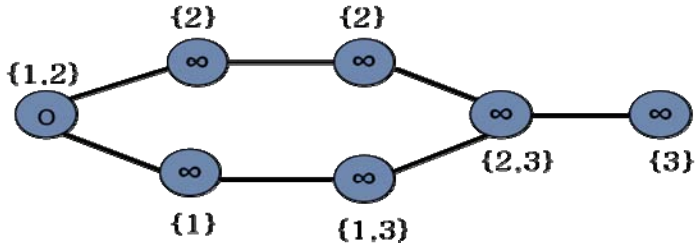

Figure: 2- Initialization of the network

The initialization of the node is done in step 1. First the source node 'a' and the interfaces associated with it is assigned zero i.e. d(a)=0, d(a, 1)=0 and d(a, 2)=0. The rest of the nodes are assigned ∞ i.e. d(b) = d(b, 1) = d(c) = d(c, 1) = d(c, 3) = d(d) = d(d, 2) = d(d, 3) = d(e) = d(e, 2) = d(f) = d(f, 1) = d(f, 2) = d(f, 3) = d(g) = d(g, 3) = ∞.

Step 2:

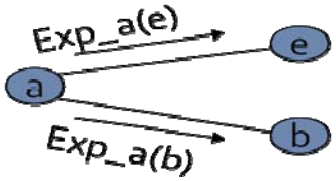

Figure 3: Sending of Explore message

After the initialization process the source node ‘a’ will send explore messages to its directly connected neighboring nodes ‘e’ and ‘b’. The explore message will have information about the sending node, receiving node and the interfaces available at the source node. Thus the explore message send by ‘a’ to node ‘b’ can be formulated as Exp_a(b) = { a, (1,2), b} and to node ‘e’ is given as Exp_a(e) = { a, (1,2), e}. Node ‘a’ is then considered as the parent node to reach ‘b’ and ‘e’.

Step 3:

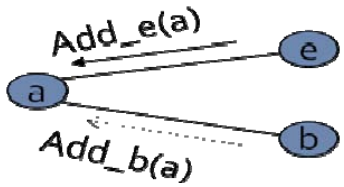

Figure 4: Sending of Add Message

After receiving the explore message, node ‘e’ and ‘b’ will look for a common interface with parent node ‘a’. If there is a common interface the nodes will send add message to ‘a’ otherwise they will reject the explore message send by ‘a’. In this case, interface 1 is common between both nodes ‘a’ and ‘b’. Then ‘b’ will send and add message, Add_b(a) = {*b*, π (*a*, 1), 1.5}, containing information about its parent, common interface and cost of activating the common interface. In this way all the nodes are added in to the network.

Step 4:

After the nodes are added into the network, the algorithm will calculate cost for each node in the graph. This is done by the relaxation process. During relaxation process, each node will check if the new cost is less than the current cost. If this condition is found to be true, then the current cost is replaced with the new cost. In the given graph, the current value of node ‘b’ and ‘e’ is infinity. The values of nodes are determined by the relaxation process. The following steps show the relaxation procedures of nodes in detail.

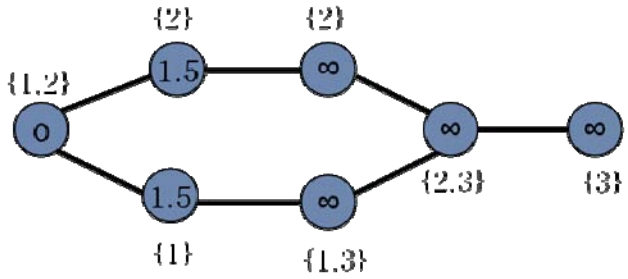

Figure 5: Relaxation Process

d (b,1) > d(a,1) +1.5 → ∞ > 0 + 1.5 (checking the current value of ‘b’)
d (b,1) = d(a,1) +1.5 → ∞ = 1.5 (assigning the cheapest value to ‘b’)
Parent node (b, 1) = ‘a’.
The value of node ‘b’, accessed by interface 1, is updated from ∞ to 1.5 and the parent node is given as ‘a’. The relaxation process is further carried on with node ‘e’.
d (e,2) > d(a,2) +1.5 → ∞ > 0 + 1.5
d (e,2) = d(a,2) +1.5 → ∞ = 1.5
Parent node (e, 2) = ‘a’.
Here the cost of node ‘e’ activated through interface 2 is updated from ∞ to 1.5 and the parent node is given as ‘a’. The relaxation process is continued with the neighbors of ‘b’ and ‘e’.

Step 5:

After the nodes find their cost of energy consumption using the relaxation process, it will inform its neighbors and update the routing table. This is done by the broadcast message.

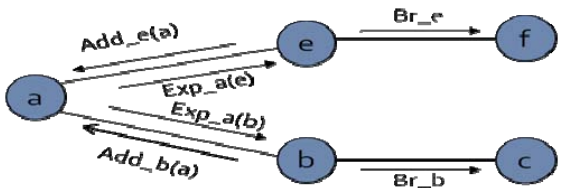

Figure 6: Flow of messages

Step 6:

The distributive iterative process proceeds further with remaining nodes 'f' , 'c', 'd' and 'g' . The relaxation process is continued and the estimated cost value is found for all the nodes and the associated interfaces. The loop will continuously run for $|V|-1$ times, where V is the number of nodes in the network. In this example, the loop terminates after 6^{th} execution of the cycle. After the 6^{th} cycle, the cost of activating each node through different interfaces is calculated and the routing table is updated with the cheapest cost found for each node.

Step 7:

After finding the best and cheapest cost for each node and co-coordinating interfaces, we have to find the cheapest path to the destination. The cheapest path algorithm is used to find the cost effective way to reach from node 'a' to 'g'. The algorithm compares the value of each node activated through different interfaces and finds the minimum cost path.

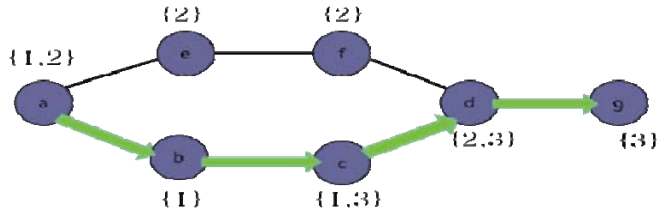

Figure 7: Obtained Cheapest Path

The figure shows the example graph we have illustrated before. The green arrow shows the cheapest path from node 'a' to 'g' after the execution of the algorithm. The cheapest path algorithm compares the cost and finds the cheapest path CP_g as a – b – c – d – g. At the end the communication is achieved from 'a' to 'g' by activating the cheapest interfaces. From the above illustration, it is evident that the algorithm works for networks with multiple interfaces and finds the cheapest path.

Table 1 summarizes the cost of activating the interfaces in the downward-path of the simple undirected network graph which we used for the algorithm analysis.

Table 1: summary of downward-path interface activation cost

No. of Iterations	Nodes Visited	Interface Activated	Minimum Cost (Interface activated)	Total Cost (At each Iteration)
1	{a}	(1)	1.5	1.5
2	{a, b}	(1,1)	1.5	3.0
3	{a,b,c}	(1, 1, 1)	1.5	4.5
4	{a, b, c, d }	(1,1,1, 3)	1.0	5.5
5	{a, b, c, d}	(1,1,1,3)	1.0	6.5
6	{a,b,c,d,g}	(1,1,1,3,3,3)	1.0	7.5

Table 2 summarizes the cost of activating the interfaces in the upper-path of the simple undirected network graph which we used for the algorithm analysis.

Table 2: summary of upper-path interface activation cost

No. of Iterations	Nodes Visited	Interface Activated	Minimum Cost (Interface activated)	Total Cost (At each Iteration)
1	{a}	(2)	1.5	1.5
2	{a, e}	(2,2)	1.5	3.0
3	{a,e,f}	(2, 2, 2)	1.5	4.5
4	{a, e, f, d }	(2,2,2, 2)	1.5	6.0
5	{a, e, f, d}	(2,2,2,2,3)	1.0	7.0
6	{a,e,f,d,g}	(2,2,2,2,3,3)	1.0	8.0

VII. Conclusion and Future work:

The distributed settings of the proposed algorithm were based on the general distributed version of the bellman-ford shortest path algorithm. The algorithm was mathematically analyzed with a network graph model and the results were provided. The proposed distributed algorithm will be the stepping stone for the future highly efficient cheapest path algorithms in the distributed setting.

One of our dream research projects in future will be extending the same cheapest path problem in context to multi-interface networks in the field of robotics and artificial intelligence. In future robots will be used for military raids instead of human beings, so that inter-robot communication will be the main issue. Robots are gathering the information about any particular task with the help of various in-built sensors.

In inter-robot communication each individual robot should transmit or receive the gathered sensor information to each other with the help of wireless interfaces such as Bluetooth, Wi-Fi, Infrared etc. The selection of the specific wireless interface by the robot should be low power consuming. The main aim is to develop a highly efficient algorithm which can be used to determine the selection of specific low power consuming wireless interfaces so that the robot will be able to switch on or off the interface depending upon the particular task and the environment.

ACKNOWLEDGEMENT

The authors wish to thank Dr. Leonardo Mostarda, Principal Lecturer in Middlesex University Department of Engineering and Information Sciences for his strong support and guidance towards completion of the work and also Dr. Purav Shah for providing the initial guidance which gave us the basic idea for the problem implementation.

REFERENCES

[1] Adrian Kosowski , Alfredo Navarra , Cristina Pinotti "Exploting multi-interface networks : Connectivity and Cheapest Paths, LLC,2009
[2] Ferruccio Barsi, Alfredo Navarra, Cristina Pinotti "Cheapest paths in Multi-interface networks, University of Perugia, 2007
[3] Andrew .C. Hoffman, "Multiple approaches for distributed algorithms", 2004
[4] Kosowski, Alfredo Navarra: "*Cost Minimization in Unbounded Multi-Interface Networks"*, In 20th PPAM workshop on (SPC), 2007.
[5] Bertossia, Alfredo Navarra, Pinotti," *Maximum Bandwidth Broadcast in Single and Multi-Interface Networks"*, 5th International conference on ubiquitous information management and communication (ICUMC), 2010.
[6] Hoang Lan Nguyen, Uzen Trang Nguyen,- *Algorithms for Bandwidth efficient Multicast Routing in Multi-Channel and Multi Radio Wireless Mesh Networks*, Department of Computer Science and Engineering, York University, Canada, 2011.
[7] Greg N. Freireckson,- *A distributed shortest path Algorithm for a planar network*, Purdue University, Department of science, 1985.

AUTHOURS PROFILE:

M.S.HARISH has completed his Master degree in Telecommunication Engineering from Middlesex University, United Kingdom in the year 2011. His research areas of interest are *wireless communications, optimization algorithms and signal processing.*

ANJANA PRASAD has completed her Master degree in Telecommunication Engineering from Middlesex University, United Kingdom in the year 2011. Presently she is working as a graduate engineer at Ford Motor in the Research and Product Development Department in Greater-London, United Kingdom. Her research areas of interest are *wireless communications, satellite communication and sensor networks.*