Methodological Aspects of Simulation and Modeling

Project Report

Modeling Routing in Wireless Ad Hoc Networks using Cell-DEVS

Umar Farooq
Student No. 100258695
SCE, Carleton University
December 1st, 2003

Table of Contents

List of Figures

Note: Additionally, there are 49 figures related to testing. They are however, not numbered and hence not included in the list of figures.

Modeling Routing in Wireless Ad Hoc Networks using Cell-DEVS

1. INTRODUCTION

1.1 Brief Description

The system chosen to be modeled with Cell-DEVS involves routing in wireless ad-hoc networks. The system models a wireless ad hoc network consisting of a number of nodes spread randomly on a plane. A node wants to send message to a particular node whose location is unknown, using the shortest path available. Cell-DEVS modeling would be used to find out the shortest path between two nodes in the ad hoc network plane. This would make use of the Lee Algorithm discussed in [1].

Each node can communicate to all the nodes on its top, down, left and right. However, each neighbor of the node may not be another node. It may be a dead cell through which communication cannot take place. This dead cell represents a physical obstacle (such as a high rise building) or simply the absence of node. Thus the communicating nodes will have to find their way around that obstacle (dead cell).

Figure 1 shows the brief sketch of system (taken from [1]).

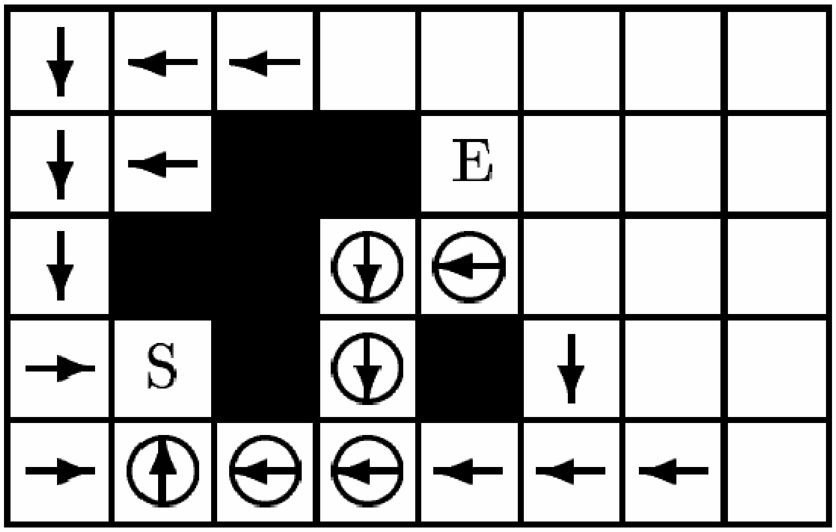
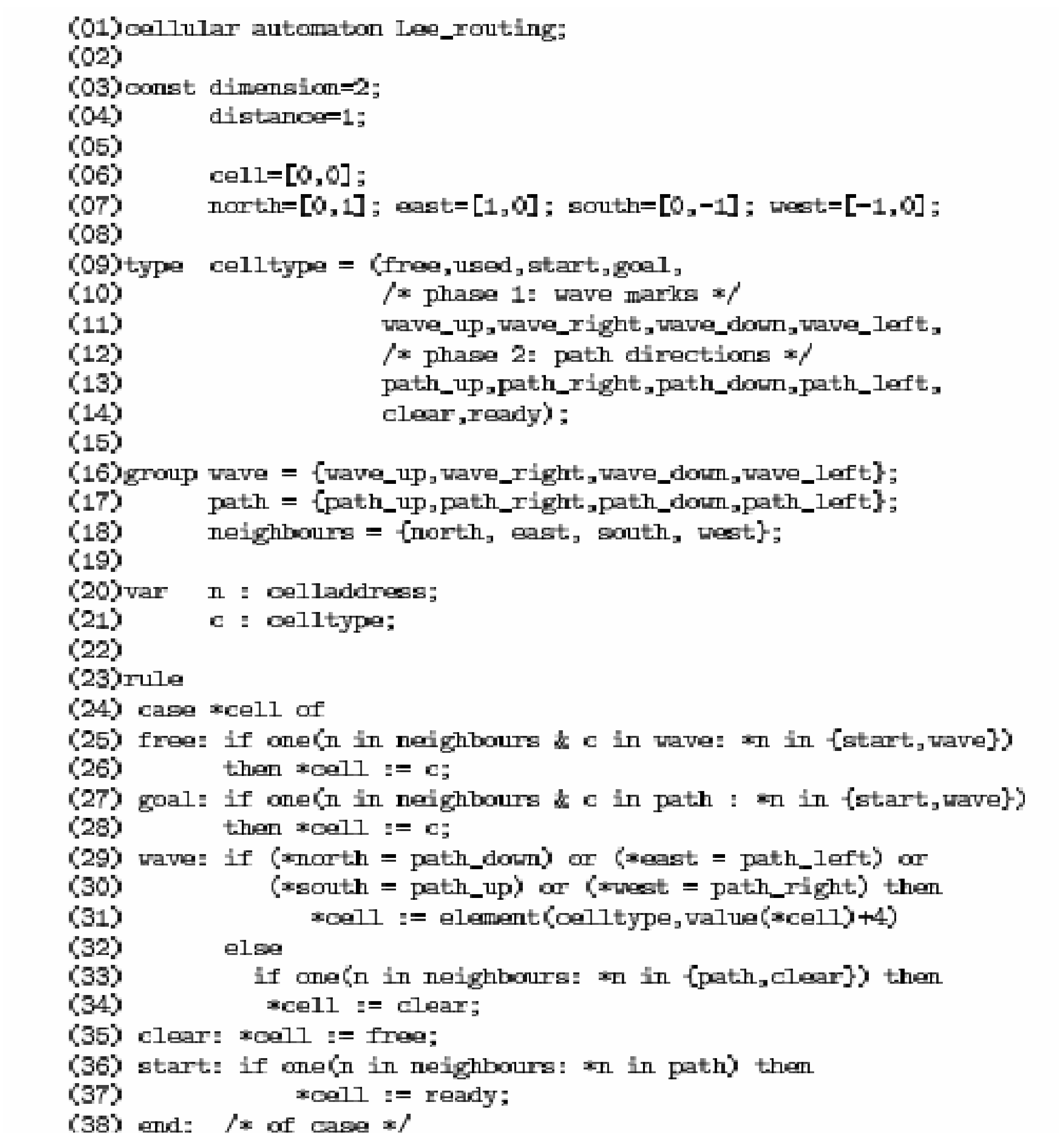


Figure 1.1: Brief Sketch of the System

Here node S represents the sender node that wants to communicate with node E but does not know its location or shortest path to it. The black nodes represent dead cell through which communication is not possible. Note that there may be more than one path from S to E but the goal of the Cell-DEVS models is to come up with the shortest path.

1.2 Lee's Algorithm

Lee's Algorithm involves 14 possible states of each of the node. The algorithm is given as follows [taken from 1].



```
cellular automaton Lee_routing;

const dimension=2;
      distance=1;

      cell=[0,0];
      north=[0,1]; east=[1,0]; south=[0,-1]; west=[-1,0];

type  celltype = (free,used,start,goal,
                 /* phase 1: wave marks */
                 wave_up,wave_right,wave_down,wave_left,
                 /* phase 2: path directions */
                 path_up,path_right,path_down,path_left,
                 clear,ready);

group wave = {wave_up,wave_right,wave_down,wave_left};
      path = {path_up,path_right,path_down,path_left};
      neighbours = {north, east, south, west};

var   n : celladdress;
      c : celltype;

rule
 case *cell of
 free: if one(n in neighbours & c in wave: *n in {start,wave})
       then *cell := c;
 goal: if one(n in neighbours & c in path : *n in {start,wave})
       then *cell := c;
 wave: if (*north = path_down) or (*east = path_left) or
          (*south = path_up) or (*west = path_right) then
             *cell := element(celltype,value(*cell)+4)
       else
         if one(n in neighbours: *n in {path,clear}) then
          *cell := clear;
 clear: *cell := free;
 start: if one(n in neighbours: *n in path) then
            *cell := ready;
 end:  /* of case */
```

Figure 1.2: Lee's Algorithm

The algorithm shows that during the first phase of the algorithm, all the nodes broadcast wave messages to their neighbors forming a reverse path to the starting node. When the message reaches the destination node, the wave node nearest to the destination becomes the path node. All the wave nodes that see a path node in their neighbor and pointing towards them also become the path node. All other wave nodes are cleared. Further details of the algorithm are available in [1] and will not be presented here to save space.

2. PROBLEM STATEMENT

The system discussed in Section 1 was successfully modeled with Cell-DEVS in an earlier work [2] (Assignment 2) using 15 states for each node to find out the shortest path between the two communicating nodes. The original algorithm by Lee involves 14 states but during experimentation it was found that it is necessary to have another state in order to make sure that each node that is not going to become the path from S to E discards the routing message that it received. A number of examples have been successfully run involving as many as 1000 nodes.

The earlier work (Assignment 2) shows that the work can be extended in a number of ways to enhance the capabilities of the model. This project extends this work in the following 3 ways.

2.1 Part 1: Intra-Network and Inter-Network Routing in 3 Dimensions

In the first part, the project extends the routing algorithm to 3 dimensions i.e. it would find the shortest path between two nodes in 3 dimensional space. Here the 3rd dimension may represent different networks (such as the Internet) to which the given ad hoc networks connects. Thus the intra-network routing in Assignment 2 is extended to inter-network routing in the project. This required an addition of 4 more states of the node to the 15 states defined in Assignment 2.

2.2 Part 2: Construction of *Optimal* Multicast Trees

As a second part, the project studies the construction of an *optimal* multicast tree where a single node sends a message to more than one node. Construction of a multicast tree for more than one receiving nodes is not a trivial task. Hochberger [1] work shows that for 8 receiver nodes, we may require as many as 768 states for each cell. Obviously these many numbers of states are beyond practical limits, especially if the number of receiver nodes is increased further. So the project modifies the original algorithm and introduces the notion of trees to make multicast trees possible with much lesser number of required states. Another aspect that has been taken into account during the construction of multicast trees is its optimality i.e. it should duplicate the same message as less as possible. Consider for example the distribution of nodes shown in Figure 3.

Here S represents the sender that wants to multicast a data to two nodes R1 and R2. For the sake of simplicity dead links (cells) are not shown in this example and hence a simple shortest path between S and R1 exists marked by 1 in each cell. Note that for R2, there exist two paths both of which involve least number of hops i.e. 8 hops from S to R2. Path 1 takes the data first to the left of S and then up to R2 while Path 2 takes that data first to the top of S and then left to R2. Note that if the same data is being multicast by S to the two nodes R1 and R2, it would make much more sense to send the date to R2 through path 2. This is because in this way the data would be duplicated only for the last 4 hops from S to R2. On the other hand if the data is sent to R2 through path 1 it is duplicated right in the beginning (i.e. 8 hops). If there are multiple nodes, least duplicating the data would save huge bandwidth. The construction of multicast trees in this project hence takes this feature into account and tries to optimize the trees for minimum bandwidth usage as much as possible.

							1	1	**R1**			
							1					
							1					
							1					
							1					
							1					
			R2	2	2	2	1,2					
			2				1,2					
			2				1,2					
			2				1,2					
			2	2	2	2	**S**					

Figure 2.1: Optimal Multicast Trees Concept

2.3 Part 3: Routing Among Multiple Pairs of Senders and Receivers

The work in Assignment 2 shows that the Lee's Algorithm fails if there are multiple pairs of senders and receivers. Hochberger [1] has shown that if there are multiple pairs of senders and receivers, the algorithm may generate deadlocks and may prevent the generation of routing path between pairs of nodes that can communicate. He has given an example of 2 pairs of nodes and has shown how to work around such problems. However, if there are a large number of pairs of senders and receivers, it becomes virtually impossible to work around that problem. As a 3rd part, the project studies this problem in greater depth. The project employs multiple planes where each plane corresponds to one state variable. The state variable in each plane is assigned to each pair of communicating nodes such that each of the possible 15 states of that variable corresponds to only a particular pair. This way multiple pairs of senders and receivers are handled without generating deadlocks or requiring to define more states.

3. IMPLEMENTATION AND ANALYSIS

3.1 Part 1: Intra-Network and Inter-Network Routing in 3 Dimensions

3.1.1 Conceptual Model Description

The conceptual model for this part has been described in Section 1.1 and Section 2.1. It will not be repeated here to save space.

3.1.2 Formal Specifications for the Cell-DEVS Model

In order to compute the shortest path between two nodes in a 3 dimensional space, a Cell-DEVS model named *path* is defined. Its formal specifications are given below.

CD = $< X, Y, I, S, \theta, N, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$

X = S
Y = S

I = $<\eta, \mu, P^x, P^y>$
Where

η = 7
μ = 0
P_j^i = $\{ (N_j^i, T_j^i) / \forall j \in [1, 7], N_j^i \in [i_1, i_7]$ and $T_j^i \in I_i \}$, $I_i = \{ x / x \in X$ if $i = X \}$ or $I_i = \{ x / x \in Y$ if $i = Y \}$;

S = $\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18\}$

θ = { (s, phase, σqueue, σ) /
s $\in$ S is the state value for a cell (S has already been defined),
phase $\in$ {active, passive},
$\sigma queue = \{ ((v_1,\sigma_1),...,(v_m,\sigma_m)) / m \in N \wedge m <\infty) \wedge \forall (i \in N, i \in [1,m]), v_i \in S \wedge \sigma_i \in R_0^+ \cup \infty\}$; and
$\sigma \in R_0^+ \cup \infty$
} (As Transport delay is used)

N = $\{(0,0,1), (-1,0,0), (0,-1,0), (0,0,0), (0,1,0), (1,0,0),(0,0,-1)\}$

d = 100

D = $\theta \times N \times d \rightarrow R_0^+ \cup \infty$,

For δ_{int}, δ_{ext}, λ and τ see Figure 1.2 and path.ma (These functions actually constitute the whole algorithm that has been given in Figure 2 and implemented in path.ma. These functions are hence not written here to save space).

GCC = $<$ Xlist, Ylist, I, X, Y, η, N, {f, c, b}, C, B, Z, select $>$

Xlist = { Ø }
Ylist = { Ø }

I = $< P^x, P^y >$
Where

P^x	=	{ Ø }
P^y	=	{ Ø }

X	=	S (Defined Above)
Y	=	S (Defined Above)
η	=	7
N	=	{(0,0,1), (-1,0,0), (0,-1,0), (0,0,0), (0,1,0), (1,0,0),(0,0,-1)}
f	=	Different values are used for different tests e.g. 5, 25, 30 etc.
c	=	Different values are used for different tests e.g. 5, 25, 30 etc.
b	=	Different values are used for different tests e.g. 5, 25, 30 etc.
C	=	$\{C_{ijk} / i \in [1,f] \wedge j \in [1,c] \wedge k \in [1,b]\}$
B	=	No-wrapped

Z =

$P_{ijk}{}^{Y_1} \rightarrow P_{i,j-1,k}{}^{X_1}$	$P_{i,j+1,k}{}^{Y_1} \rightarrow P_{ijk}{}^{X_1}$
$P_{ijk}{}^{Y_2} \rightarrow P_{i+1,j,k}{}^{X_2}$	$P_{i-1,j,k}{}^{Y_2} \rightarrow P_{ijk}{}^{X_2}$
$P_{ijk}{}^{Y_3} \rightarrow P_{i,j+1,k}{}^{X_3}$	$P_{i,j-1,k}{}^{Y_3} \rightarrow P_{ijk}{}^{X_3}$
$P_{ijk}{}^{Y_4} \rightarrow P_{i-1,j,k}{}^{X_4}$	$P_{i+1,j,k}{}^{Y_4} \rightarrow P_{ijk}{}^{X_4}$
$P_{ijk}{}^{Y_5} \rightarrow P_{i,j,k}{}^{X_5}$	$P_{i,j,k}{}^{Y_5} \rightarrow P_{ijk}{}^{X_5}$
$P_{ijk}{}^{Y_6} \rightarrow P_{i,j,k+1}{}^{X_6}$	$P_{i+1,j,k+1}{}^{Y_6} \rightarrow P_{ijk}{}^{X_6}$
$P_{ijk}{}^{Y_7} \rightarrow P_{i,j,k-1}{}^{X_7}$	$P_{i,j,k-1}{}^{Y_7} \rightarrow P_{ijk}{}^{X_7}$

Select = { (-1,0,0), (1,0,0), (0,0,0), (0,1,0), (0,-1,0), (0,0,1), (0,0,-1) }

3.1.3 Test Strategy

The model will be tested by having different initial distributions of the node and checking whether the algorithm successfully determines the shortest path between two nodes (if one exists). The initial distribution of the nodes would be provided as an input to the model in terms of .map files. The .map file provided as an input to the model should have the following characteristics.

- There is only one start point.
- There is only one end point.
- There is at least one start point.
- There is at least one end point.

The first two conditions are necessary because the Lee's algorithm in its current form fail if there are multiple senders or receivers. The last two points are the requirements of the problem statement. Note that there may no path exists between the starting and the end point. In such a situation, the system would end up with some nodes in the wave conditions and the other nodes in initial state. If more than one path exists, the system would locate the shortest path.

3.1.4 Implementation in CD++

For implementation of the algorithm in CD++, the 15 states defined in Assignment 2 were designated the following state values. Moreover, for the third dimension another 4 states were added. These 19 states along with their state values are given as follows.

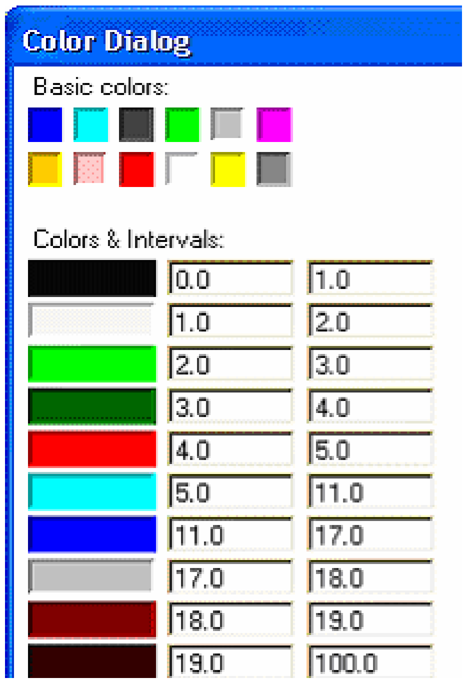
State 0	Dead Cell (Broken Communication Link)
State 1	Initial State of the Nodes
State 2	Initial State of the Destination Node
State 3	Destination Ready (State of the Destination Node after it has received a send request from the sender)
State 4	Initial State of the Sender Node
State 5	Wave Up ↑
State 6	Wave Down ↓
State 7	Wave Right →
State 8	Wave Left ←
State 9	Wave in positive 3rd dimension ↗
State 10	Wave in negative 3rd dimension ↙
State 11	Path Up ↑⃝
State 12	Path Down ↓⃝
State 13	Path Right →⃝
State 14	Path Left ←⃝
State 15	Path in positive 3rd dimension ↗⃝
State 16	Path in negative 3rd dimension ↙⃝
State 17	Clear State (Final state of the nodes that received a wave message but are not going to become the path)

State 18 Destination Found (Final State of the Sender Node).

Using these state values the model was implemented in CD++. The path.ma file is presented next. Note here that in [path-rule] many Boolean statements could have been combined together. However, for the sake of clarity each statement is written in a separate line. The *initialMapValue* has been defined as path.map. This is the name of the file that the model takes as an input. Transport delay is used and a no-wrapped border is used. The dimensions of the model given here are for a particular example. Different tests were run with different dimensions.

```
[top]
components : path

[path]
type : cell
dim : (5,5,5)
delay : transport
defaultDelayTime : 100
border : nowrapped
neighbors :             path(0,0,1)
neighbors :             path(-1,0,0)
neighbors : path(0,-1,0)  path(0,0,0)  path(0,1,0)
neighbors :             path(1,0,0)
neighbors :             path(0,0,-1)
initialvalue : 1
initialMapValue : path.map
localtransition : path-rule

[path-rule]
rule : 3 100 { (0,0,0) = 2 and (stateCount(15) > 0 or stateCount(16)> 0
      or stateCount(11) > 0 or stateCount(12) > 0 or stateCount(13) > 0
      or stateCount(14) > 0)}

rule : 5 100 { (0,0,0) = 1 and (-1,0,0) > 3 and (-1,0,0) < 11}
rule : 6 100 { (0,0,0) = 1 and (1,0,0) > 3 and (1,0,0) < 11}
rule : 7 100 { (0,0,0) = 1 and (0,1,0) > 3 and (0,1,0) < 11}
rule : 8 100 { (0,0,0) = 1 and (0,-1,0) > 3 and (0,-1,0) < 11}
rule : 9 100 { (0,0,0) = 1 and (0,0,1) > 3 and (0,0,1) < 11}
rule : 10 100 { (0,0,0) = 1 and (0,0,-1) > 3 and (0,0,-1) < 11}

rule : 11 100 { (0,0,0) = 5 and stateCount(2) = 1}
rule : 12 100 { (0,0,0) = 6 and stateCount(2) = 1}
rule : 13 100 { (0,0,0) = 7 and stateCount(2) = 1}
rule : 14 100 { (0,0,0) = 8 and stateCount(2) = 1}
rule : 15 100 { (0,0,0) = 9 and stateCount(2) = 1}
rule : 16 100 { (0,0,0) = 10 and stateCount(2) = 1}

rule : 11 100 { (0,0,0) = 5 and (0,-1,0) = 13}
rule : 11 100 { (0,0,0) = 5 and (0,1,0) = 14}
rule : 11 100 { (0,0,0) = 5 and (1,0,0) = 11}
rule : 11 100 { (0,0,0) = 5 and (0,0,1) = 16}
rule : 11 100 { (0,0,0) = 5 and (0,0,-1) = 15}

rule : 12 100 { (0,0,0) = 6 and (0,-1,0) = 13}
rule : 12 100 { (0,0,0) = 6 and (0,1,0) = 14}
rule : 12 100 { (0,0,0) = 6 and (-1,0,0) = 12}
rule : 12 100 { (0,0,0) = 6 and (0,0,1) = 16}
rule : 12 100 { (0,0,0) = 6 and (0,0,-1) = 15}

rule : 13 100 { (0,0,0) = 7 and (0,-1,0) = 13}
rule : 13 100 { (0,0,0) = 7 and (1,0,0) = 11}
rule : 13 100 { (0,0,0) = 7 and (-1,0,0) = 12}
rule : 13 100 { (0,0,0) = 7 and (0,0,1) = 16}
rule : 13 100 { (0,0,0) = 7 and (0,0,-1) = 15}

rule : 14 100 { (0,0,0) = 8 and (0,1,0) = 14}
rule : 14 100 { (0,0,0) = 8 and (1,0,0) = 11}
rule : 14 100 { (0,0,0) = 8 and (-1,0,0) = 12}
rule : 14 100 { (0,0,0) = 8 and (0,0,1) = 16}
rule : 14 100 { (0,0,0) = 8 and (0,0,-1) = 15}

rule : 15 100 { (0,0,0) = 9 and (0,-1,0) = 13}
rule : 15 100 { (0,0,0) = 9 and (0,1,0) = 14}
rule : 15 100 { (0,0,0) = 9 and (-1,0,0) = 12}
rule : 15 100 { (0,0,0) = 9 and (1,0,0) = 11}
rule : 15 100 { (0,0,0) = 9 and (0,0,-1) = 15}

rule : 16 100 { (0,0,0) = 10 and (0,-1,0) = 13}
rule : 16 100 { (0,0,0) = 10 and (0,1,0) = 14}
rule : 16 100 { (0,0,0) = 10 and (-1,0,0) = 12}
rule : 16 100 { (0,0,0) = 10 and (1,0,0) = 11}
rule : 16 100 { (0,0,0) = 10 and (0,0,1) = 16}

rule : 17 100 { (0,0,0) = 1 and stateCount(17) > 0}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and stateCount(17) > 0}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and stateCount(3) > 0}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and stateCount(18) > 0}

rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and (-1,0,0) > 10 and
      (-1,0,0) < 17 and (-1,0,0) != 12}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and (1,0,0) > 10 and
      (1,0,0) < 17 and (1,0,0) != 11}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and (0,1,0) > 10 and
      (0,1,0) < 17 and (0,1,0) != 14}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and (0,-1,0) > 10 and
      (0,-1,0) < 17 and (0,-1,0) != 13}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and (0,0,1) > 10 and
      (0,0,1) < 17 and (0,0,1) != 16}
rule : 17 100 { (0,0,0) > 4 and (0,0,0) < 11 and (0,0,-1) > 10 and
      (0,0,-1) < 17 and (0,0,-1) != 15}

rule : 18 100 { (0,0,0) = 4 and stateCount(11) > 0}
rule : 18 100 { (0,0,0) = 4 and stateCount(12) > 0}
rule : 18 100 { (0,0,0) = 4 and stateCount(13) > 0}
rule : 18 100 { (0,0,0) = 4 and stateCount(14) > 0}
rule : 18 100 { (0,0,0) = 4 and stateCount(15) > 0}
rule : 18 100 { (0,0,0) = 4 and stateCount(16) > 0}

rule : {(0,0,0)} 100 {t}
```

In order to for the results to be viewed in VRML GUI a color palette is defined. This palette is presented next as it is necessary to explain the results presented in the next section. Note here that in the palette all the *wave* states of the nodes (state 5 to state 10) are represented with the same color. Similarly, all the *path* states of the node (state 11 to state 16) are being represented by the same color.



⬛	0.0	1.0
⬜	1.0	2.0
🟩	2.0	3.0
🟩	3.0	4.0
🟥	4.0	5.0
🟦	5.0	11.0
🟦	11.0	17.0
⬜	17.0	18.0
🟫	18.0	19.0
🟫	19.0	100.0

Figure 3.1: Color Palette for Viewing Results of Part 1 in VRML GUI

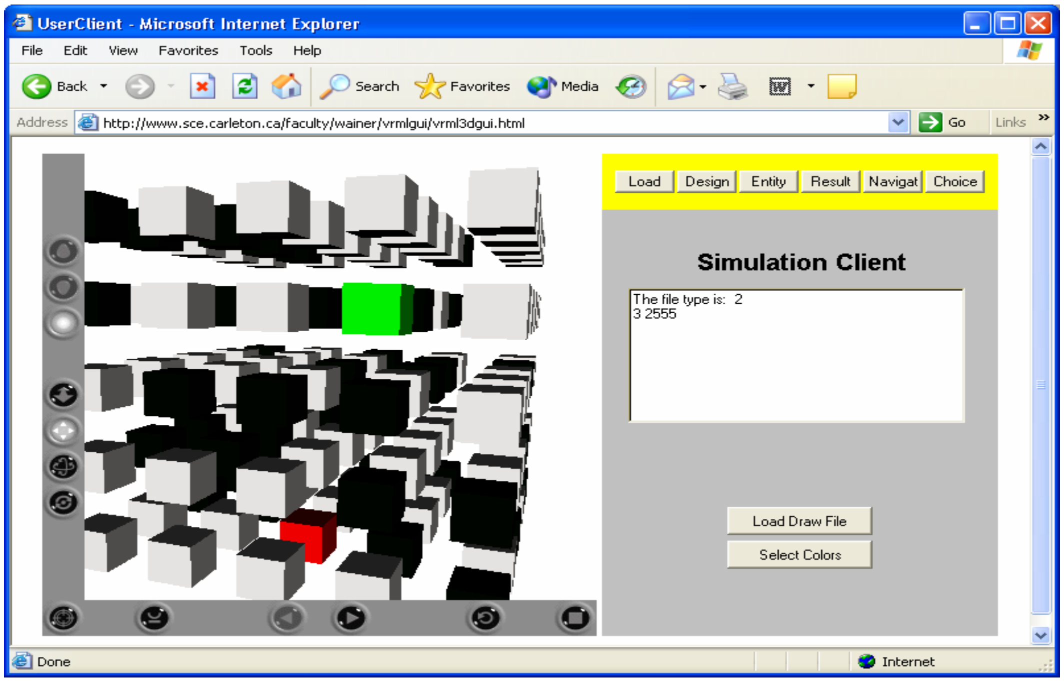
Thus all the dead cells are represented as black cells and all the cells that have not received any message yet represented as white. The destination is represented as light green but after receiving the send request from the sender it changes its color to dark green. Sender is red. All the wave messages are light blue and the path nodes dark blue. Those wave nodes that are not going to become the path gets to clear state which is represented as grey. The sender after successful discovery of the path to the destination changes its color to maroon.

3.1.5 Testing

A number of tests were conducted on the model. A subset of these tests is presented in this section. Note that these tests are given in the attached diskette. As the input of each test is a map file which is very long, it is not included in the report. Similarly, .drw files are very long, so they are not included in the report either. The screen shots taken from execution of the model on the VRML GUI at different intervals during the test are given in this report.

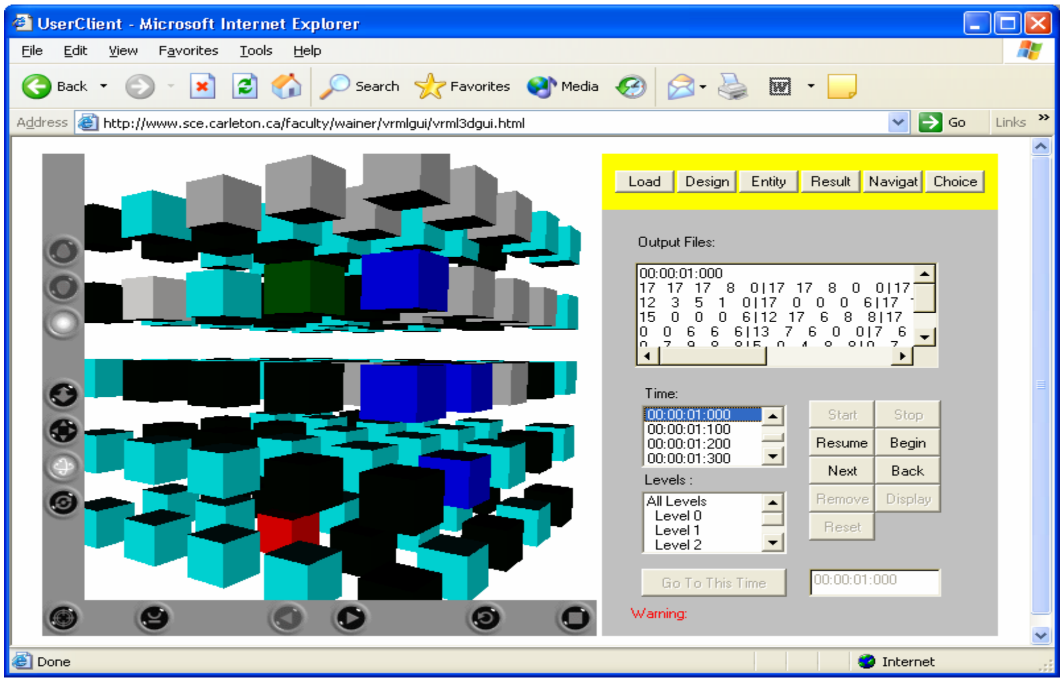
a) Test 1

This is a test consisting of 5x5x5 Cell space. The initial distribution of the nodes is given as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Simulation Client
Load Draw File
Select Colors
Done
Internet

```
The file type is:  2
3 2555
```

The state of the model after 10 steps of execution is given as follows:

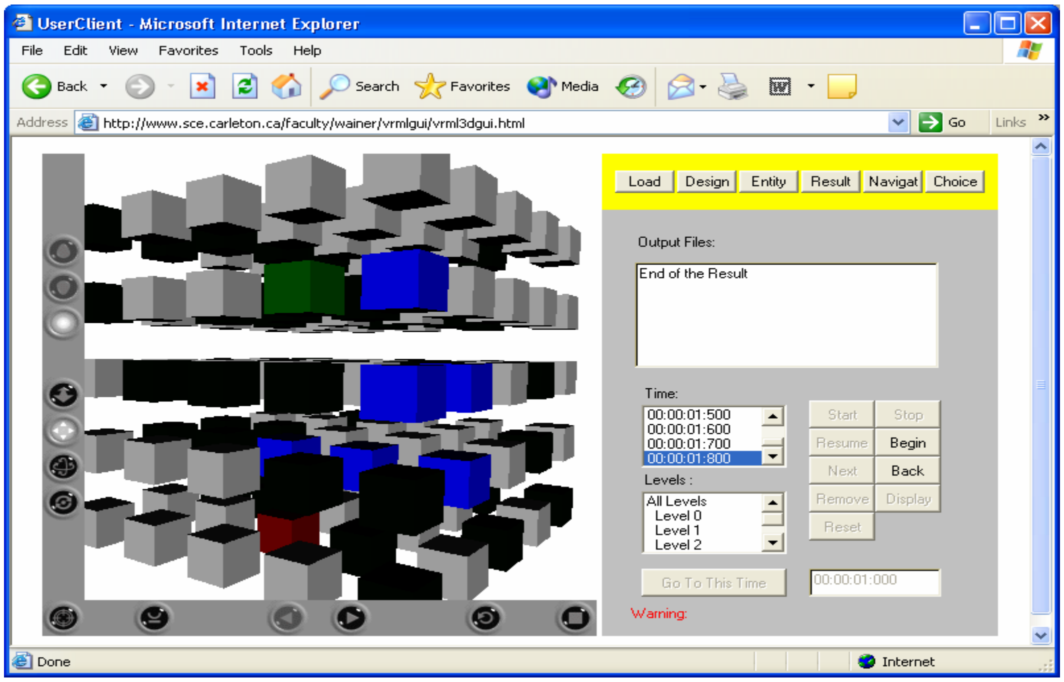

UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Output Files:
Time:
Levels :
All Levels
Level 0
Level 1
Level 2
Start Stop
Resume Begin
Next Back
Remove Display
Reset
Go To This Time
Warning:
Done
Internet

```
00:00:01:000
17  17  17   8   0|17  17   8   0   0|17
12   3   5   1   0|17   0   0   0   6|17
15   0   0   0   6|12  17   6   8   8|17
0   0   6   6   6|13   7   6   0   0|7   6
```

```
00:00:01:000
00:00:01:100
00:00:01:200
00:00:01:300
```

```
00:00:01:000
```

The final state of the model after 18 steps is as follows:



http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

End of the Result

```
00:00:01:500
00:00:01:600
00:00:01:700
00:00:01:800
```

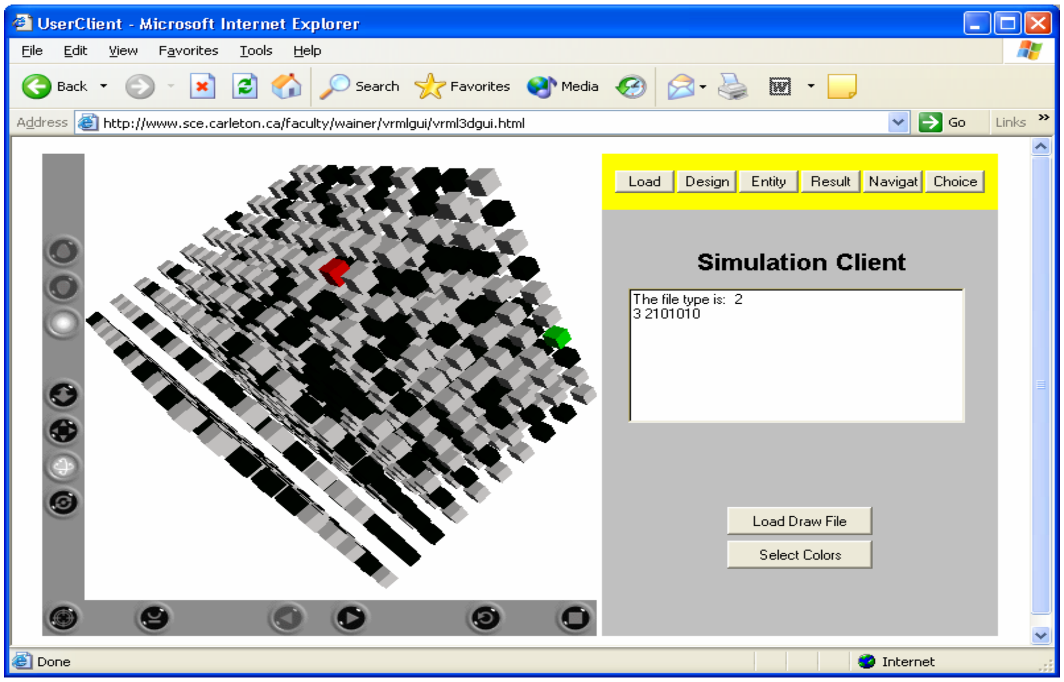
```
All Levels
Level 0
Level 1
Level 2
```

00:00:01:000

Note that the model has successfully established the shortest path between the sender and the receiver and all wave nodes end up in clear state.

b) Test 2

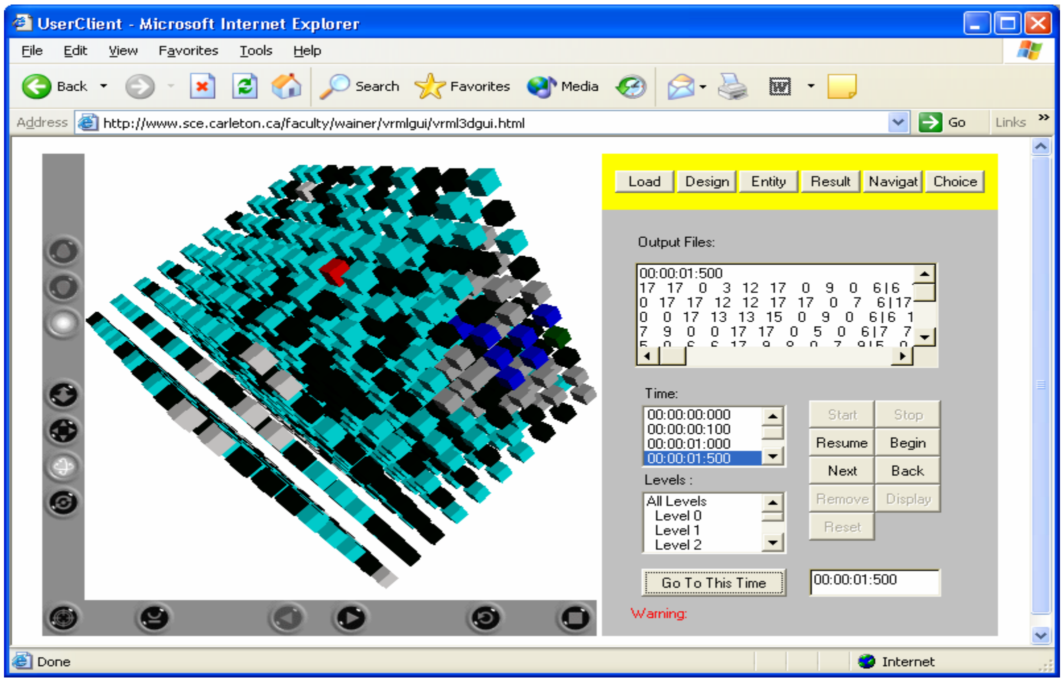
This is a test consisting of 10x10x10 Cell space. The initial distribution of the nodes is given as follows:



http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

```
The file type is:  2
3 2101010
```

The state of the model after 15 steps of execution is given as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Done
Internet

Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

Output Files:

```
00:00:01:500
17  17   0   3  12  17   0   9   0   6|6
0  17  17  12  12  17  17   0   7   6|17
0   0  17  13  13  15   0   9   0   6|6  1
7   9   0   0  17  17   0   5   0   6|7   7
```

Time:

```
00:00:00:000
00:00:00:100
00:00:01:000
00:00:01:500
```

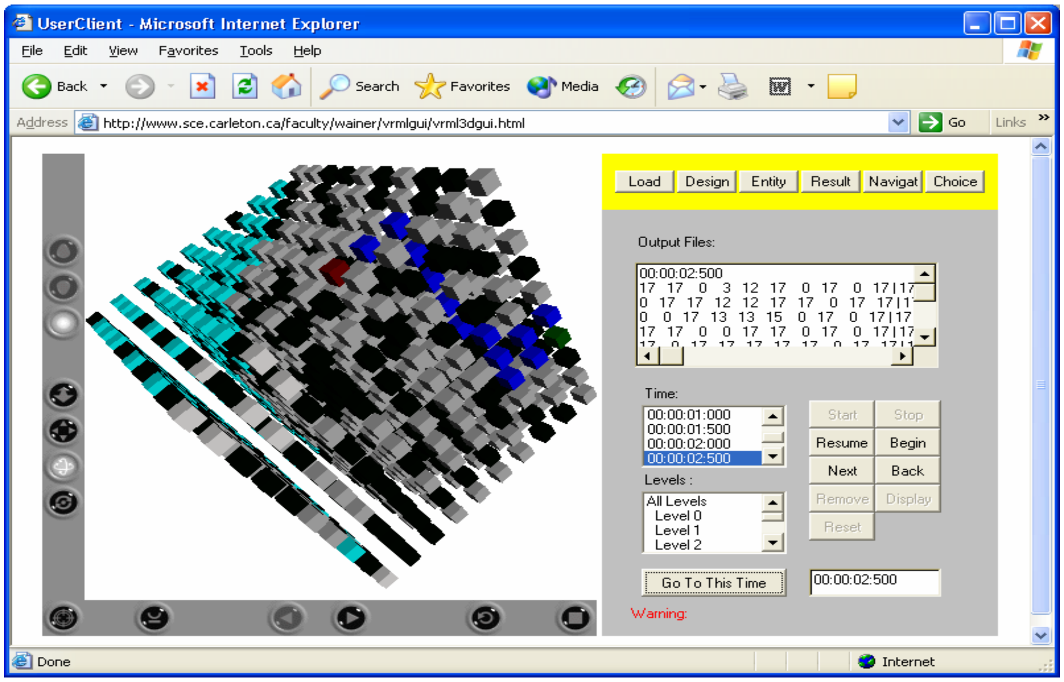
Levels :

```
All Levels
Level 0
Level 1
Level 2
```

00:00:01:500

Warning:

The state of the model after 25 steps of execution is given as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Done
Internet

Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

Output Files:

```
00:00:02:500
17  17   0   3  12  17   0  17   0  17|17
0  17  17  12  12  17  17   0  17  17|1
0   0  17  13  13  15   0  17   0  17|17
17  17   0   0  17  17   0  17   0  17|17
```

Time:

```
00:00:01:000
00:00:01:500
00:00:02:000
00:00:02:500
```

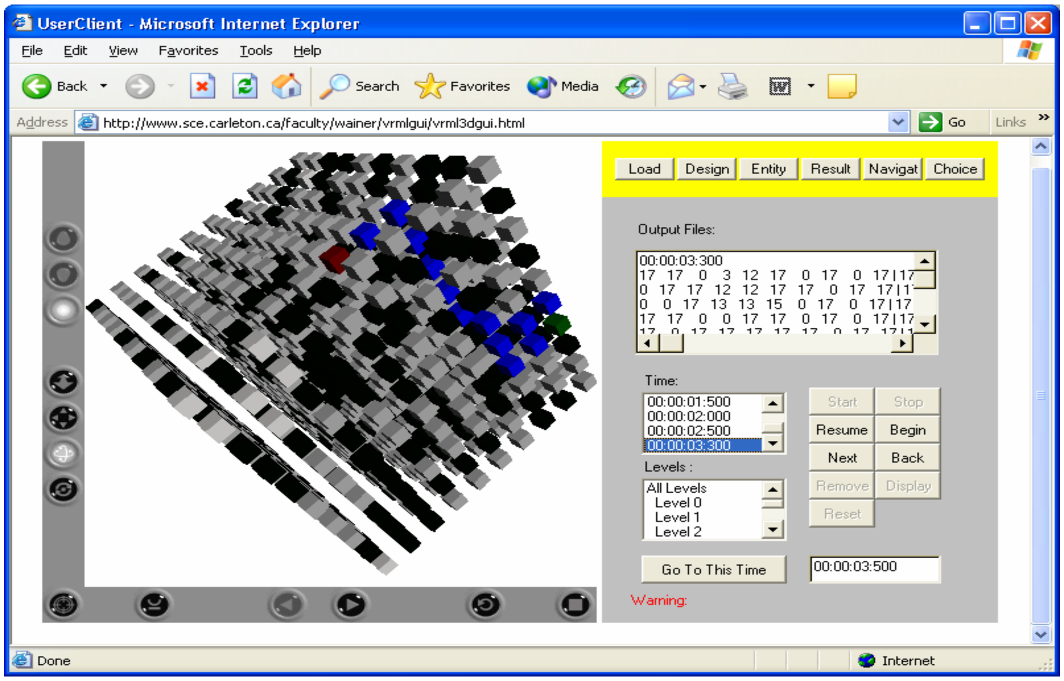
Levels :

```
All Levels
Level 0
Level 1
Level 2
```

00:00:02:500

Warning:

The final state of the model after 33 steps is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Done
Internet

Output Files:

```
00:00:03:300
17  17   0   3  12  17   0  17   0  17|17
0  17  17  12  12  17  17   0  17  17|1
0   0  17  13  13  15   0  17   0  17|17
17  17   0   0  17  17   0  17   0  17|17
```

Time:

```
00:00:01:500
00:00:02:000
00:00:02:500
00:00:03:300
```

Levels :

```
All Levels
  Level 0
  Level 1
  Level 2
```

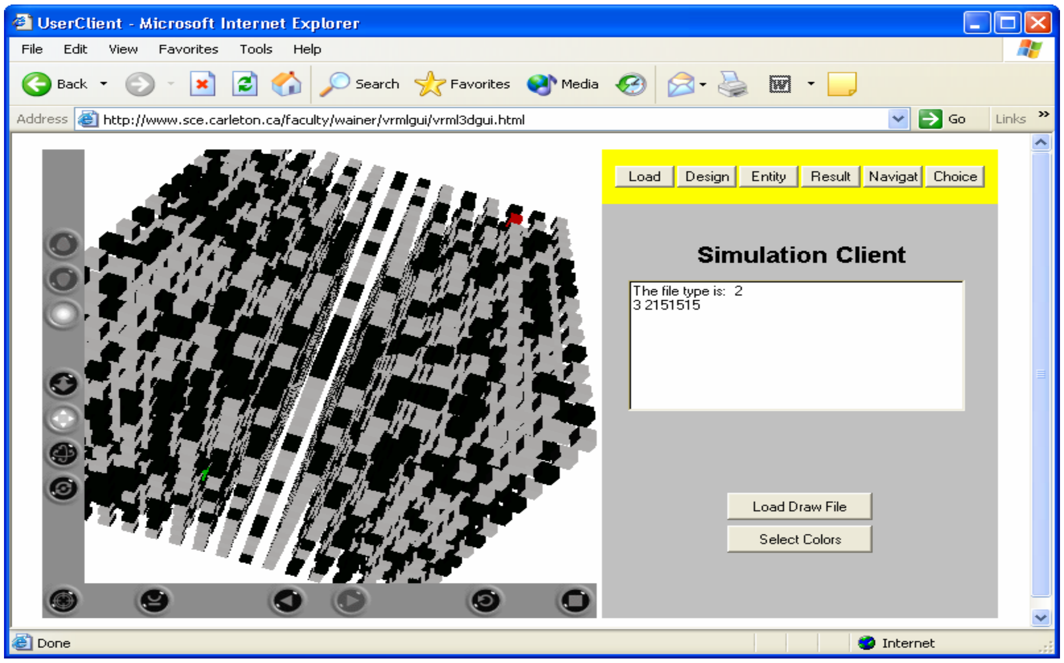
00:00:03:500

Warning:

Note that the model has successfully established the shortest path between the sender and the receiver and all wave nodes end up in clear state. Note that near the receiver, there are two possible paths shown in dark blue: one from above of the receiver and the other from just front of the receiver. Both of these paths represent the shortest path between the sender and the receiver.

c) Test 3

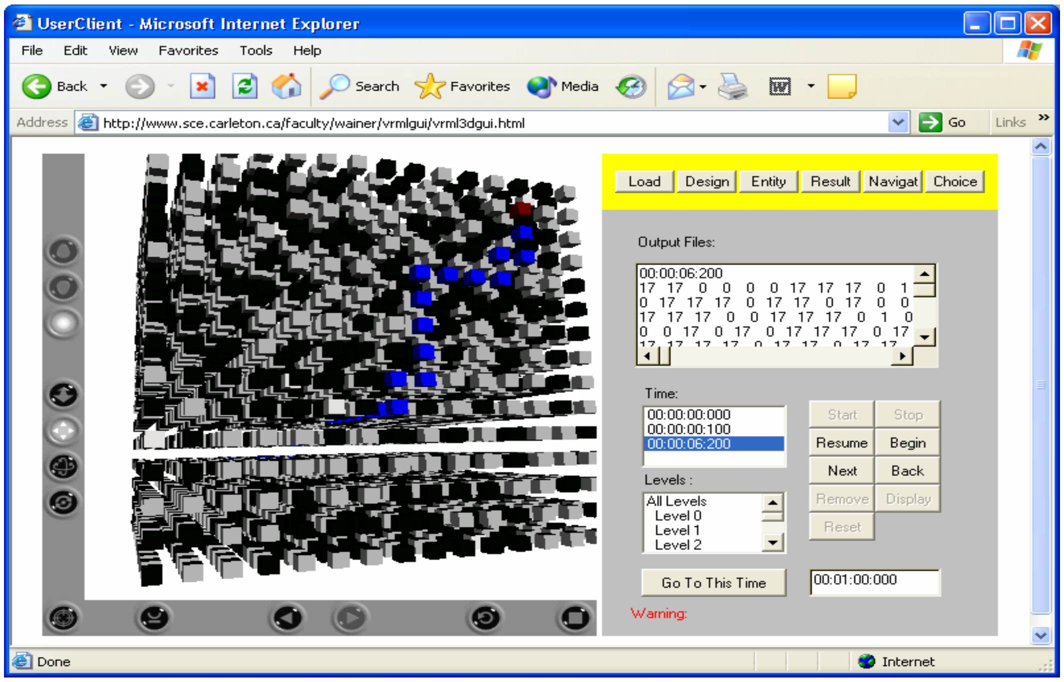
This is a test consisting of 15x15x15 Cell space. The initial distribution of the nodes is given as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Load Design Entity Result Navigat Choice
Load Draw File
Select Colors
Done
Internet

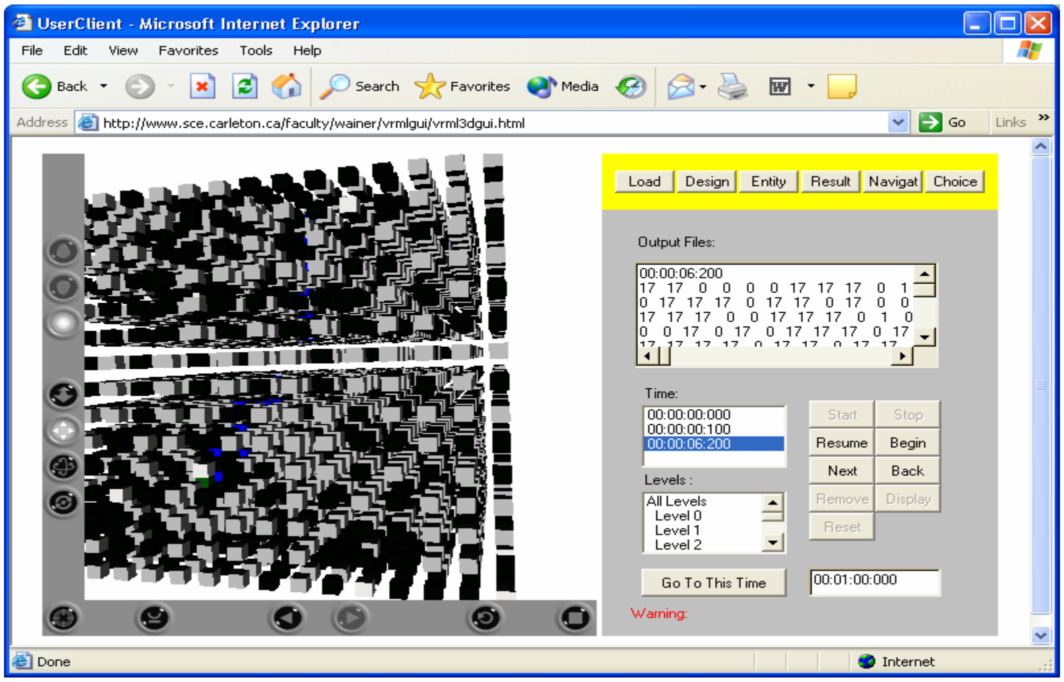
Simulation Client

```
The file type is:  2
3 2151515
```

The final state of the model after 62 steps is as follows:


```
00:00:06:200
17  17   0   0   0   0  17  17  17   0   1
0  17  17  17   0  17  17   0  17   0   0
17  17  17   0   0  17  17  17   0   1   0
0   0  17   0  17   0  17  17  17   0  17
```

From another angle:



```
00:00:06:200
17  17   0   0   0   0  17  17  17   0   1
0  17  17  17   0  17  17   0  17   0   0
17  17  17   0   0  17  17  17   0   1   0
0   0  17   0  17   0  17  17  17   0  17
```

Note that the model has successfully established the shortest path between the sender and the receiver and all wave nodes end up in clear state. As can be seen from the above tests as the number of nodes increases, it becomes increasingly difficult to view the results in the VRML GUI. Thus, tests having more nodes than in Test 3 are not presented here.

3.1.6 Reaction of the Model to Different Inputs than Those Defined in the Specifications

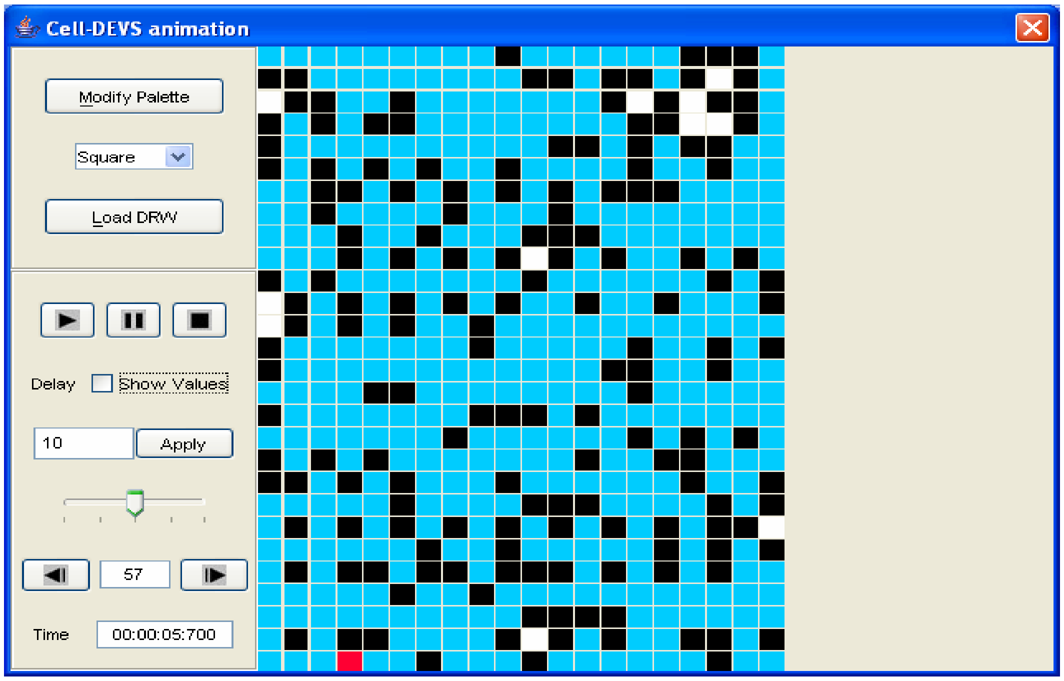
The requirement of the model is that it has one and only one sender and one and only one receiver. Now let us consider one by one all the options that violate this condition.

a) No Sender Node

If there is no sender node, then the final state of the model would be the same as its initial state. This is because it is the sender node that initiates the request for the path. If none of the nodes in the model see a sender node they would not change their state. Hence the final state of the model would be the same as its initial state.

b) No Receiver Node

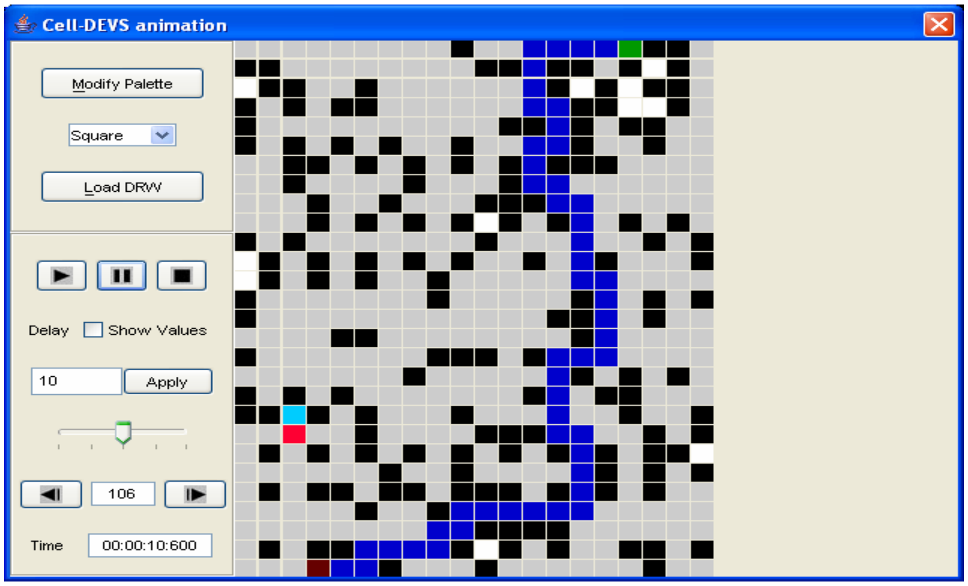
If there is no receiver node, then all the nodes that can receive wave messages would receive that message and the system would end up with nodes in the wave state. This is because unless any of the nodes see a receiver, they don't form a path and until the path is formed none of the nodes in wave state go into clear state. For the sake of simplicity and greater visualization let us take an example of a plane (instead of 3-dimensional space) of nodes in which there is no receiver node. The final state of the model after 57 steps would be like the one shown as follows.



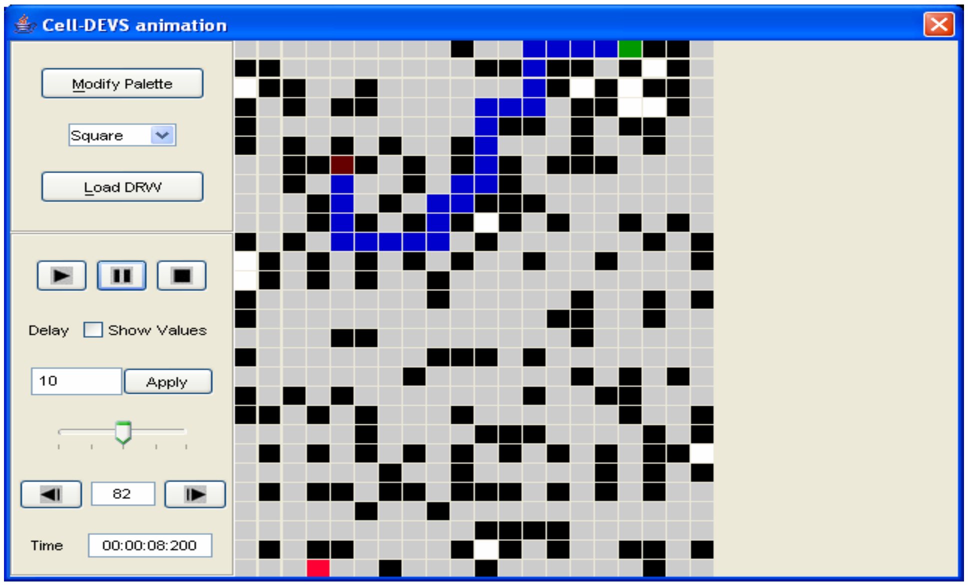
The test with 3 dimensions would have shown a similar behavior.

c) Multiple Sender Nodes

If there are multiple sender nodes then the model can behave in an unexpected manner. This is because the wave states for one of the sender nodes may interfere with the path for other sender node or else prevent the formation of the wave and path. For the sake of simplicity and greater visualization let us take an example of a plane (instead of 3-dimensional space) in which there are two sender nodes, the final state of the model after 106 steps would be like the one shown below.



Here the path has been successfully established for one of the sender nodes while there is no path for the other sender node. However, if we change the position of the sender node the model behaves as follows.

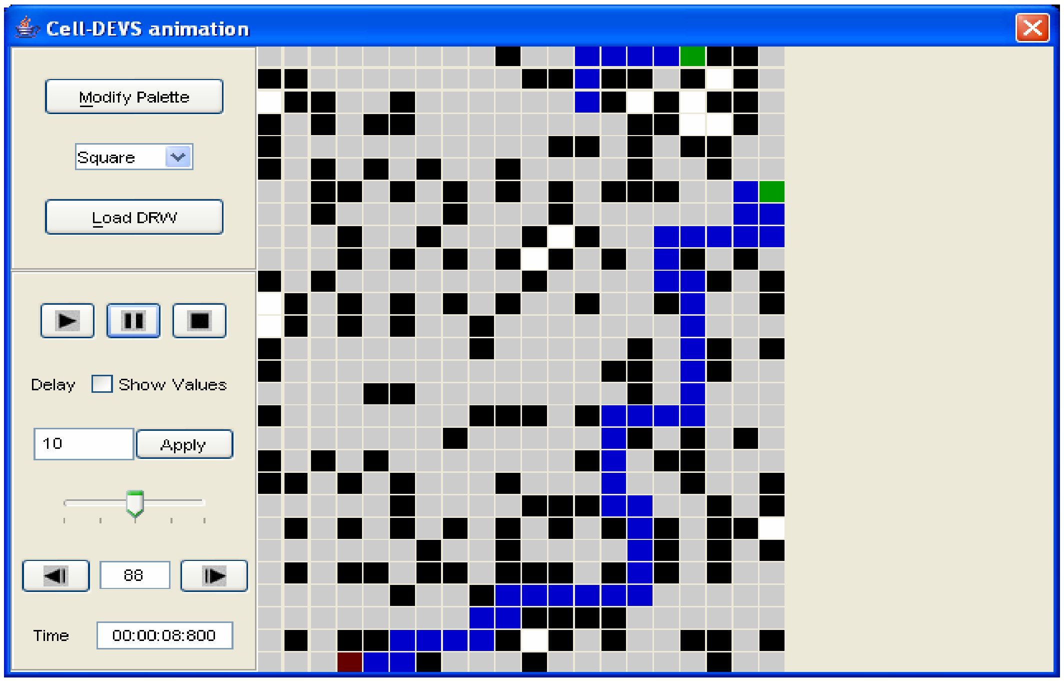


Note that in this test, the model formed a path for one of the sender nodes. This clearly shows that the wave states for the lower receiver interfered with the wave states for the upper receiver. The test in 3 dimensions shows a similar behavior.

The conclusion is thus that the model behaves in an unexpected manner with multiple senders and hence should be run with one and only one sender.

d) Multiple Receiver Nodes

If there are multiple receiver nodes then the model can behave in an unexpected manner. This is because once the model has reached the first receiver it starts sending the clear state message. Nodes that become clear cannot become a path for the second receiver. So the model usually ends up with a successful shortest path between the sender and the receiver that is nearest to sender and unsuccessful path attempt between the sender and the other receiver. For the sake of simplicity and greater visualization let us take an example of a plane (instead of 3-dimensional space) in which there are two receiver nodes. The above mentioned behavior has been observed in the following example.



The test in 3 dimensions shows a similar behavior.

The conclusion is thus that the model may behave in an unexpected manner with multiple receivers and hence should be run with one and only one receiver.

3.2 Part 2: Construction of *Optimal* Multicast Trees

3.2.1 Conceptual Model Description

The conceptual model for this part has been described in Section 1.1 and Section 2.2. It will not be repeated here to save space.

3.2.2 Algorithm Design for Optimal Multicast Trees Construction

As discussed in Section 2.2, the original Lee's algorithm breaks for multiple receiver and sender nodes. So a new algorithm was designed to make the construction of optimal multicast trees possible. This is explained stepwise in this section.

- During the first step of the algorithm, just like the Lee's algorithm, the nodes that see a sender node in their neighborhood start pointing towards it. All other nodes that see a wave node in their neighborhood start pointing towards them. The process continues till one of the wave nodes sees a receiver in its neighborhood. That node becomes the path node. All other wave nodes that see a path node in their neighborhood pointing towards them also becomes a path. The wave nodes that either do not have a path node in their neighborhood or have a path node in their neighborhood but not pointing towards them, gets to clear state. Thus, after the successful completion of the first step, a path is established between the sender and the receiver node (if one exists) and all the wave messages are purged.

- During the second step of the algorithm all the path nodes become the tree nodes and hence a tree is formed between the sender and the receiver. Moreover, during this phase, all clear state nodes are re-initialized to their initial value.

- During the third step, an input is given to the model. This input tells the model that another node in the model wants to join the multicast tree. Note here that this input should be given to the model only after the successful completion of step 2. Otherwise, the messages for the next steps may interact with the messages from first and second step and hence produce an undesirable behavior.

- During the fourth step a path is established between the new node and the nearest tree node. This is done by following the exact procedure in step 1. The only difference is that instead of a particular receiver node, the path is formed between the new node and the nearest tree node.

- Since the wave messages are broadcasted in fourth step and there are more than one tree nodes in the model, the fourth step may generate more than one path between the tree and the new node. During the fifth step all such undesirable paths are purged by sending out a clear state message and also by adding logic to the model to detect and purge non-optimal paths.

- The successful completion of the fifth step generates a shortest, optimal path from the new node to the tree. This path becomes the tree during this step. Moreover, during this phase, all clear state nodes are re-initialized to their initial value.

- After the successful completion of the sixth step, all steps from 3 to 6 are repeated for each additional node. Thus, the algorithm can add as many nodes to the multicast trees as desired, by repeating the procedure from step 3 to 6.

3.2.3 Formal Specifications for the Cell-DEVS Model

In order to make optimal multicast trees, a Cell-DEVS model named *path* is defined. Its formal specifications are given below.

CD = $< X, Y, I, S, \theta, N, d, \delta_{int,} \delta_{ext,} \tau, \lambda, D >$

X = S
Y = S

I = $<\eta, \mu, P^x, P^y>$
Where
η = 5
μ = 0
P_j^i = $\{ (N_j^i, T_j^i) / \forall j \in [1, 5], N_j^i \in [i_1, i_5]$ and $T_j^i \in I_i$ $\}, I_i = \{ x / x \in X$ if $i = X \}$ or $I_i = \{ x / x \in Y$ if $i = Y \}$;

S = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17}

θ = { (s, phase, σqueue, σ) /
s $\in$ S is the state value for a cell (S has already been defined),
phase $\in$ {active, passive},
σqueue = $\{ ((v_1,\sigma_1),...,(v_m,\sigma_m)) / m \in N \wedge m <\infty) \wedge \forall (i \in N, i \in [1,m]), v_i \in S \wedge \sigma_i \in R_0^+ \cup \infty\}$; and
$\sigma \in R_0^+ \cup \infty$
} (As Transport delay is used)

N = {(-1,0), (0,-1), (0,0), (0,1), (1,0)}

d = 100

D = $\theta \times N \times d \rightarrow R_0^+ \cup \infty$,

For δ_{int}, δ_{ext}, λ and τ see path.ma (These functions actually constitute the whole algorithm that has been discussed in Section 3.2.2 and implemented in path.ma. These functions are hence not written here to save space).

GCC	=	< Xlist, Ylist, I, X, Y, η, N, {f, c}, C, B, Z, select >
Xlist	=	{(9,3), (16,19), (23,19) } //Different cells were given inputs in different tests. The numbers given here are shown only as an example (Test 2).
Ylist	=	{ Ø }
I	=	< P^x, P^y >

Where

P^x	=	{ <X(9,3), 4>, <X(16,19), 4>, >, <X(23,19), 4> } (For Test 2)
P^y	=	{ Ø }

X	=	S (Defined Above)
Y	=	S (Defined Above)
η	=	5
N	=	{({(-1,0), (0,-1), (0,0), (0,1), (1,0)}
f	=	Different values are used for different tests e.g. 5, 25, 30 etc.
c	=	Different values are used for different tests e.g. 5, 25, 30 etc.
C	=	$\{C_{ij} / i \in [1,f] \wedge j \in [1,c] \}$
B	=	No-wrapped

Z =

$P_{ij}\,Y_1 \rightarrow P_{i,j-1}\,X_1$	$P_{i,j+1}\,Y_1 \rightarrow P_{ij}\,X_1$
$P_{ij}\,Y_2 \rightarrow P_{i+1,j}\,X_2$	$P_{i-1,j}\,Y_2 \rightarrow P_{ij}\,X_2$
$P_{ij}\,Y_3 \rightarrow P_{i,j+1}\,X_3$	$P_{i,j-1}\,Y_3 \rightarrow P_{ij}\,X_3$
$P_{ij}\,Y_4 \rightarrow P_{i-1,j}\,X_4$	$P_{i+1,j}\,Y_4 \rightarrow P_{ij}\,X_4$
$P_{ij}\,Y_5 \rightarrow P_{ij}\,X_5$	$P_{ij}\,Y_5 \rightarrow P_{ij}\,X_5$

Select = { (-1,0), (1,0), (0,0), (0,1), (0,-1) }

3.2.4 Test Strategy

The model will be tested by having different initial distributions of the node and checking whether the algorithm successfully determines the shortest path between two nodes (if one exists). The initial distribution of the nodes would be provided as an input to the model in terms of .map files. The .map file provided as an input to the model should have the following characteristics.

- There is only one start point.
- There is only one end point.
- There is at least one start point.
- There is at least one end point.

The first two conditions are necessary because the Lee's algorithm in its current form fail if there are multiple senders or receivers. The last two points are the requirements of the problem statement. Note that there may no path exists between the starting and the end point. In such a situation, the system would end up with some nodes in the wave conditions and the other nodes in initial state. If more than one path exists, the system would locate the shortest path.

After the algorithm has found the shortest path between the two nodes in the .map file and hence an initial tree has been setup, additional nodes will be added to the multicast trees. These nodes would be provided as an input to the model through the use events file. Hence the model also takes path.ev file as an input. The requirement for such a file is that it should add only one sender/receiver node to the model at one time. Before adding the next sender/receiver node to the model, enough time should be given for the first node to become the part of the multicast tree and all wave messages to be purged i.e. the wave nodes to reach the clear and than initial state.

3.2.5 Implementation in CD++

For implementation of the algorithm in CD++, a total of 18 states has been defined. These 18 states along with their state values are given as follows.

State 0	Dead Cell (Broken Communication Link)
State 1	Initial State of the Nodes
State 2	Initial State of the Destination Node
State 3	Destination Ready (State of the Destination Node after it has received a send request from the sender)
State 4	Initial State of the Sender Node
State 5	Wave Up ↑
State 6	Wave Down ↓

State 7	Wave Right →
State 8	Wave Left ←
State 9	Path Up ↑
State 10	Path Down ↓
State 11	Path Right →
State 12	Path Left ←
State 13	Clear State (Intermediate State of the nodes that received a wave message but are not going to become the path).
State 14	Destination Found (Final State of the Sender Node before the construction of the tree).
State 15	Tree State (State of the nodes that belong to the multicast tree)
State 16	Sender Tree State (Final State of the Sender Node after the construction of the tree)
State 17	Receiver Tree State (Final State of the Receiver Node after the construction of the tree)

Using these state values the model was implemented in CD++. The path.ma file is presented next. Note here that in [path-rule] many Boolean statements could have been combined together. However, for the sake of clarity each statement is written in a separate line. The *initialMapValue* has been defined as path.map. This is the name of the file that the model takes as an input. Transport delay is used and a no-wrapped border is used. The width and height of the model given here are for a particular example. Different tests were run with different values of width and height. Moreover, the events receiving nodes here are for a particular test. For different tests, different nodes receive the input events (a request to become the part of the multicast tree).

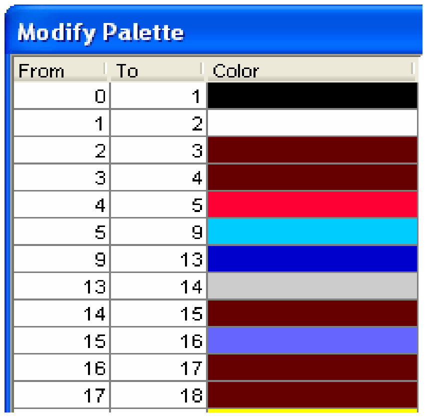
```
[top]
components : path
in : in1 in2 in3
link : in1 in1@path
link : in2 in2@path
link : in3 in3@path

[path]
type : cell
width : 20
height : 28
delay : transport
defaultDelayTime : 100
in : in1 in2 in3
border : nowrapped
```

```
link : in1 inp@path(9,3)
link : in2 inp@path(16,19)
link : in3 inp@path(23,19)
neighbors :             path(-1,0)
neighbors : path(0,-1)  path(0,0)  path(0,1)
neighbors :             path(1,0)
initialvalue : 1
initialMapValue : path.map
localtransition : path-rule
portInTransition : inp@path(9,3) special-rule
portInTransition : inp@path(16,19) special-rule
portInTransition : inp@path(23,19) special-rule

[path-rule]
rule : 3 100 { (0,0) = 2 and stateCount(9) > 0}
rule : 3 100 { (0,0) = 2 and stateCount(10) > 0}
rule : 3 100 { (0,0) = 2 and stateCount(11) > 0}
rule : 3 100 { (0,0) = 2 and stateCount(12) > 0}

rule : 5 100 { (0,0) = 1 and (-1,0) > 3 and (-1,0) < 9}
rule : 6 100 { (0,0) = 1 and (1,0) > 3 and (1,0) < 9}
rule : 7 100 { (0,0) = 1 and (0,1) > 3 and (0,1) < 9}
rule : 8 100 { (0,0) = 1 and (0,-1) > 3 and (0,-1) < 9}

rule : 9 100 { (0,0) = 5 and (stateCount(2) = 1 or stateCount(15) = 1)
      and (stateCount(9) + stateCount(10) + stateCount(11)+
      stateCount(12) = 0)}
rule : 10 100 { (0,0) = 6 and (stateCount(2) = 1 or stateCount(15) = 1)
      and (stateCount(9) + stateCount(10) + stateCount(11)+
      stateCount(12) = 0)}
rule : 11 100 { (0,0) = 7 and (stateCount(2) = 1 or stateCount(15) = 1)
      and (stateCount(9) + stateCount(10) + stateCount(11)+
      stateCount(12) = 0)}
rule : 12 100 { (0,0) = 8 and (stateCount(2) = 1 or stateCount(15) = 1)
      and (stateCount(9) + stateCount(10) + stateCount(11)+
      stateCount(12) = 0)}
rule : 9 100 { (0,0) = 5 and (0,-1) = 11}
rule : 9 100 { (0,0) = 5 and (0,1) = 12}
rule : 9 100 { (0,0) = 5 and (1,0) = 9}
rule : 10 100 { (0,0) = 6 and (0,-1) = 11}
rule : 10 100 { (0,0) = 6 and (0,1) = 12}
rule : 10 100 { (0,0) = 6 and (-1,0) = 10}
rule : 11 100 { (0,0) = 7 and (0,-1) = 11}
rule : 11 100 { (0,0) = 7 and (-1,0) = 10}
rule : 11 100 { (0,0) = 7 and (1,0) = 9}
rule : 12 100 { (0,0) = 8 and (0,1) = 12}
rule : 12 100 { (0,0) = 8 and (-1,0) = 10}
rule : 12 100 { (0,0) = 8 and (1,0) = 9}

rule : 13 100 { (0,0) > 4 and (0,0) < 9 and stateCount(13) > 0}
rule : 13 100 { (0,0) > 4 and (0,0) < 9 and stateCount(3) > 0}
rule : 13 100 { (0,0) > 4 and (0,0) < 9 and stateCount(14) > 0}
rule : 13 100 { (0,0) > 4 and (0,0) < 9 and (-1,0) > 8 and (-1,0) < 13
      and (-1,0) != 10}
rule : 13 100 { (0,0) > 4 and (0,0) < 9 and (1,0) > 8 and (1,0) < 13
      and (1,0) != 9}
```

```
rule : 13 100 { (0,0) > 4 and (0,0) < 9 and (0,-1) > 8 and (0,-1) < 13
      and (0,-1) != 11}
rule : 13 100 { (0,0) > 4 and (0,0) < 9 and (0,1) > 8 and (0,1) < 13
      and (0,1) != 12}
rule : 13 100 { (0,0) > 8 and (0,0) < 13 and (stateCount(2) +
      stateCount(3) + stateCount(4) + stateCount(5) + stateCount(6) +
      stateCount(7) + stateCount(8) + stateCount(9)+ stateCount(10)+
      stateCount(11)+ stateCount(12)+ stateCount(14) + stateCount(15)+
      stateCount(16) < 3)}
rule : 13 100 { (0,0) > 8 and (0,0) < 13 and (-1,0) > 8 and (-1,0) < 13
      and (0,0) != 9 and (-1,0) !=10}
rule : 13 100 { (0,0) > 8 and (0,0) < 13 and (0,-1) > 8 and (0,-1) < 13
      and (0,0) != 12 and (0,-1) !=11}
rule : 13 100 { (0,0) = 9 and (-1,0) = 13 }
rule : 13 100 { (0,0) = 10 and (1,0) = 13 }
rule : 13 100 { (0,0) = 11 and (0,1) = 13 }
rule : 13 100 { (0,0) = 12 and (0,-1) = 13 }

rule : 14 100 { (0,0) = 4 and stateCount(9) > 0}
rule : 14 100 { (0,0) = 4 and stateCount(10) > 0}
rule : 14 100 { (0,0) = 4 and stateCount(11) > 0}
rule : 14 100 { (0,0) = 4 and stateCount(12) > 0}

rule : 1 100 { (0,0) = 13 and (stateCount(4) + stateCount(5)+
      stateCount(6) + stateCount(7) + stateCount(8) = 0) }

rule : 15 100 { (0,0) > 8 and (0,0) < 13 and stateCount(14) > 0 }
rule : 15 100 { (0,0) > 8 and (0,0) < 13 and stateCount(15) > 0 and
      ((0,1) = 12 or (0,-1) = 11 or (-1,0) = 10 or (1,0) = 9)}
rule : 15 100 { (0,0) > 8 and (0,0) < 13 and (stateCount(3) +
      stateCount(15) > 1)}

rule : 16 100 { (0,0) = 14 }
rule : 17 100 { (0,0) = 3 and stateCount(15) > 0 }

rule : {(0,0)} 100 {t}

[special-rule]
rule : {portValue(thisPort)} 100 {t}
```

In order to for the results to be viewed in CD++ Modeler a color palette (Figure 3.2) is defined. This palette is presented next as it is necessary to explain the results presented in the next section. Note here that in the palette all the *wave* states of the nodes (state 5 to state 8) are represented with the same color. Similarly, all the *path* states of the node (state 9 to state 12) are being represented by the same color.

Here all the dead cells are represented as black cells and all the cells that have not received any message yet represented as white. As in a multicast tree any node can send or receive a message from the multicast group, from multicast tree's point of view there is no difference between the sender and the receiver. Hence all the states of the sender and the receiver are represented with the same color maroon. All the wave messages are light blue and the path nodes dark blue. Those wave nodes that are not going to become the path gets to an intermediate clear state which is represented as grey. The tree is represented in a violet color.


Modify Palette

From	To	Color
0	1	⬛
1	2	⬜
2	3	🟫
3	4	🟫
4	5	🟥
5	9	🟦
9	13	🟦
13	14	▨
14	15	🟫
15	16	🟪
16	17	🟫
17	18	🟫

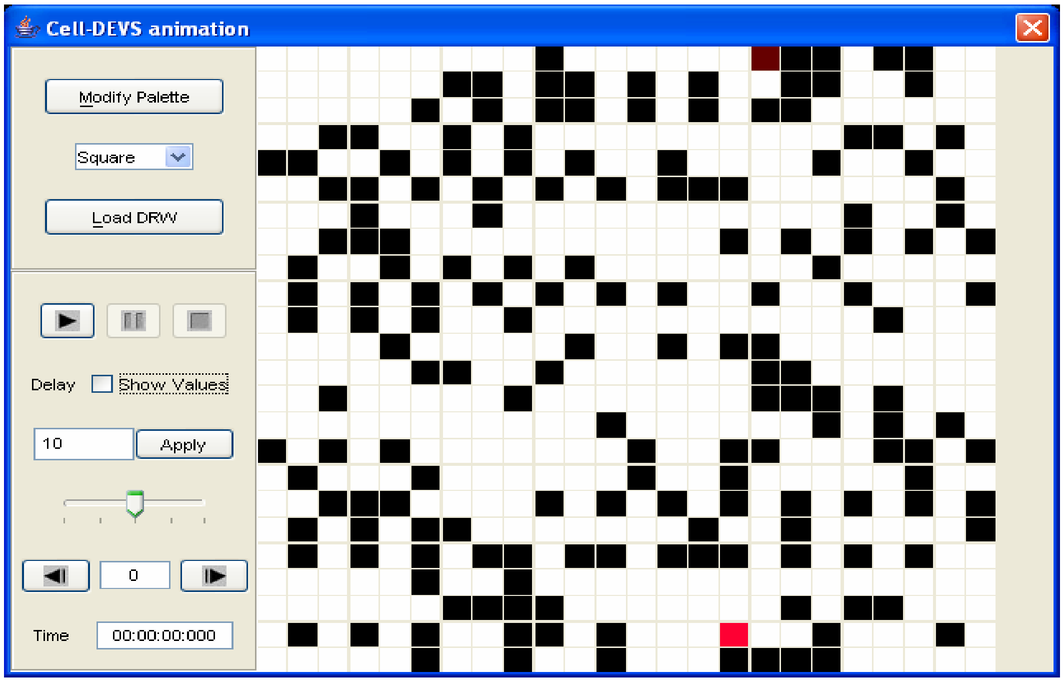
Figure 3.2: Color Palette for Viewing Results of Part 2 in CD++ Modeler

3.2.6 Testing

A number of tests were conducted on the model. A subset of these tests is presented in this section. Note that these tests are given in the attached diskette. As the input of each test is a map file which is very long, it is not included in the report. Similarly, .drw files are very long, so they are not included in the report either. The screen shots taken from execution of the model on the CD++ Modeler at different intervals during the test are given in this report.

a) Test 1

This is a test consisting of 25x25 Cell plane. Initially there are two nodes. Subsequent nodes are added after tree for each of the nodes is completed. The initial distribution of the nodes is given as follows:



Square

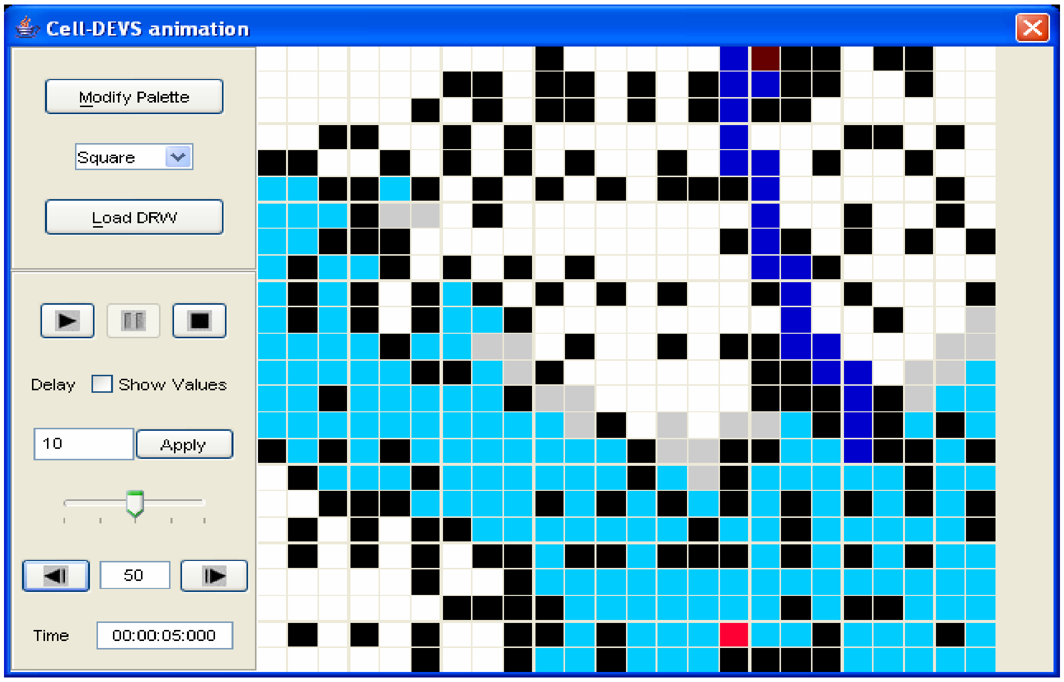
Delay ☐ Show Values

10

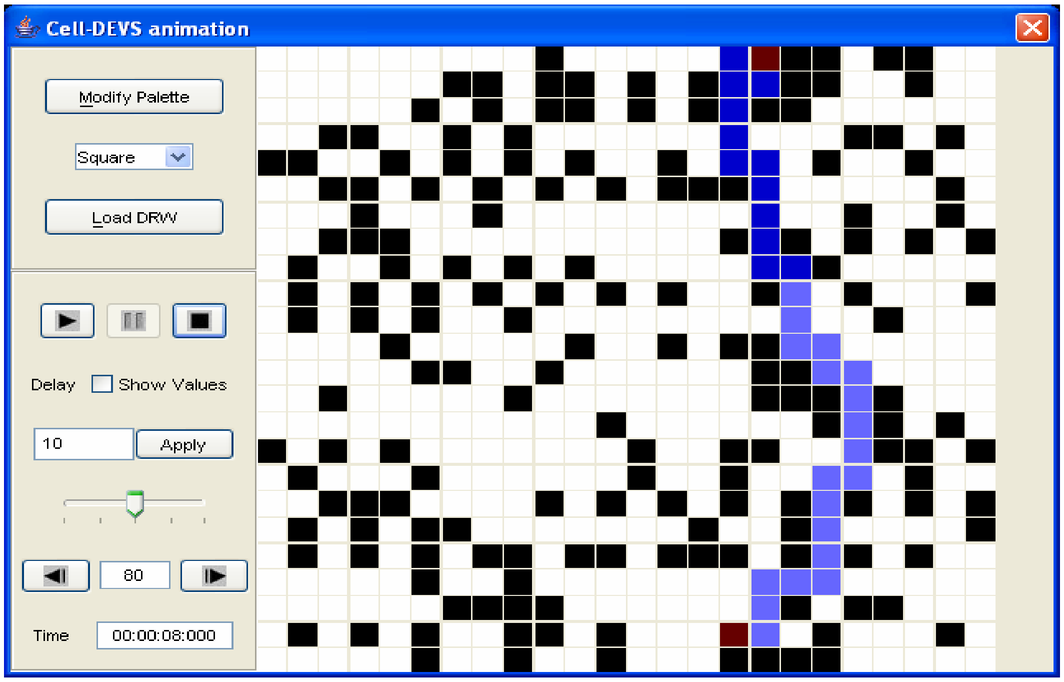
0

Time 00:00:00:000

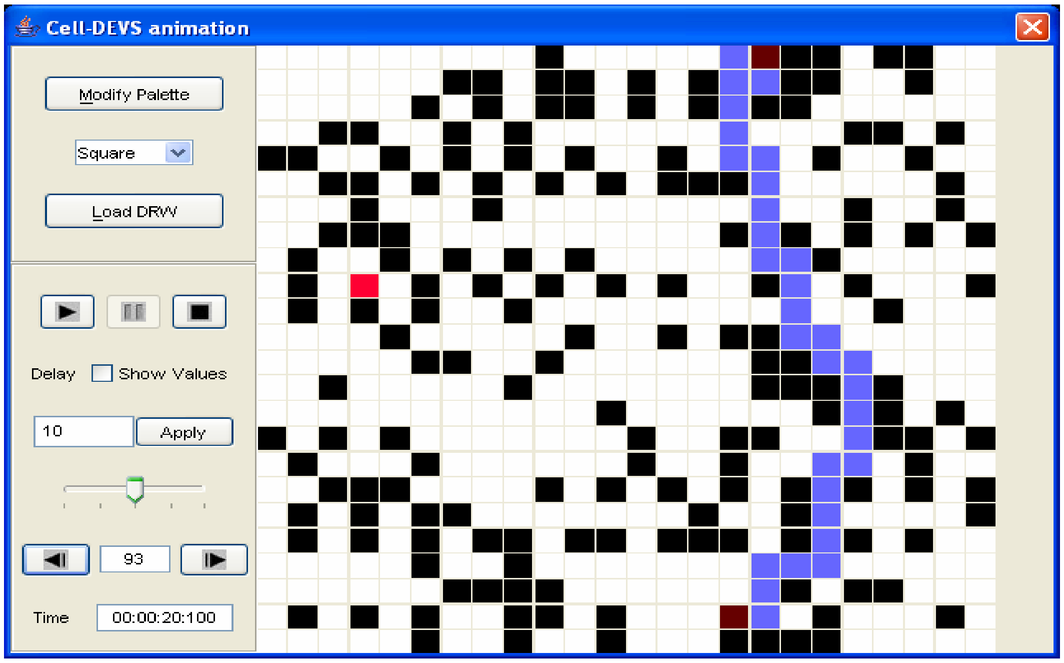
The state of the model after 50 steps of execution is as follows. Here the path between the two nodes is being formed. Also note that all the wave nodes are being set to clear state and the clear state nodes are being re-initialized.



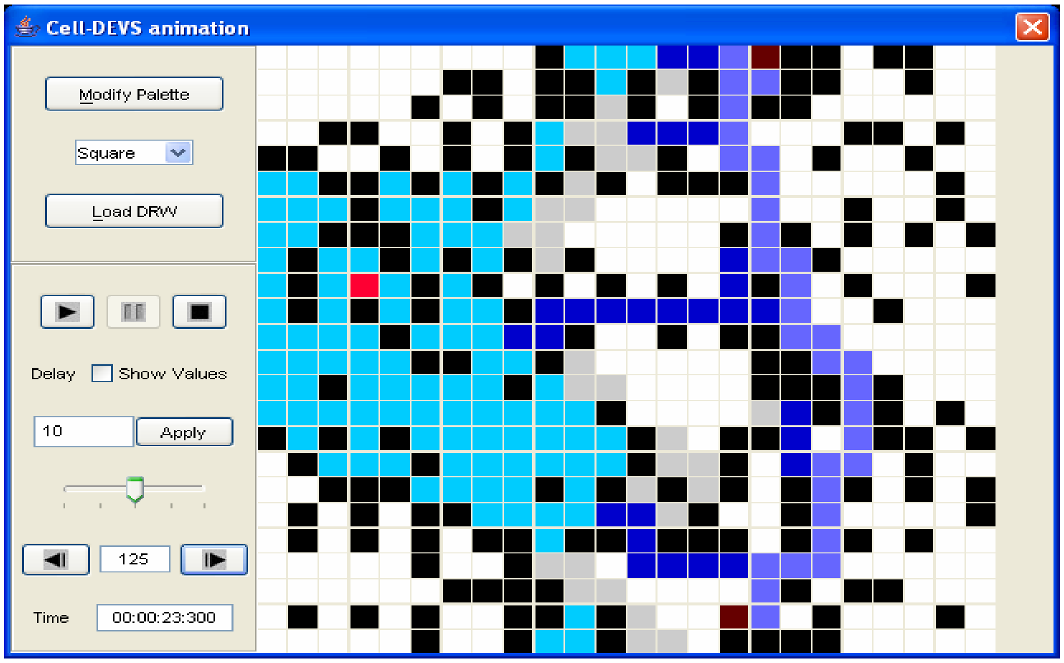
The state of the model after 80 steps of execution is as follows. Here the path between the two nodes has been completed and that path is becoming the tree. All clear state nodes have been re-initialized.



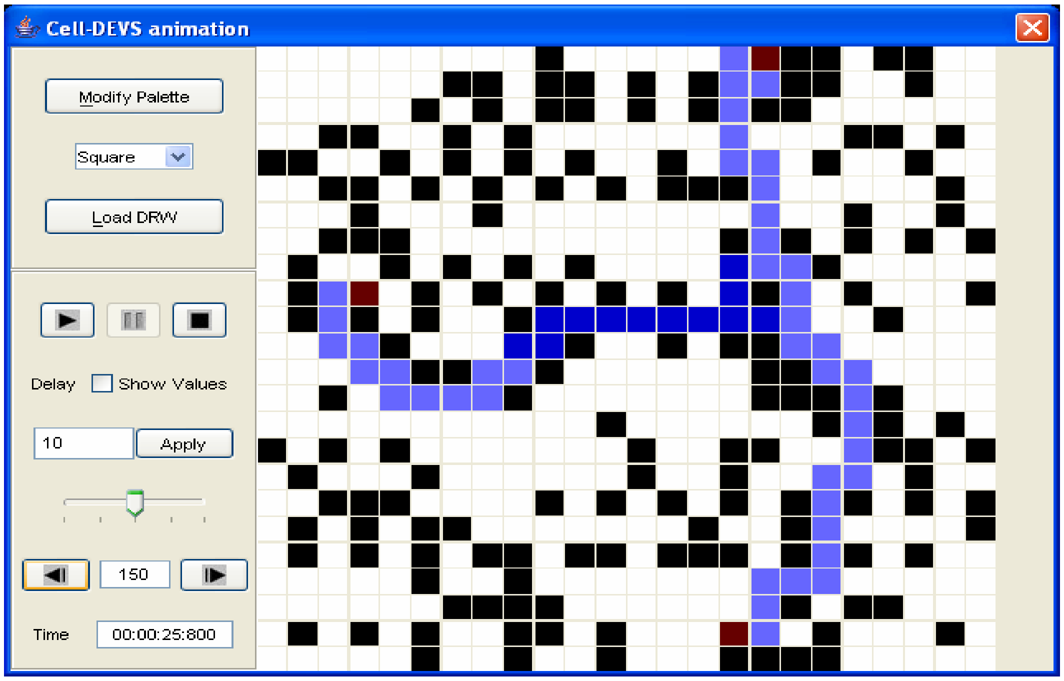
The state of the model after 93 steps of execution is as follows. Here the tree between the two nodes has been completed and a new node has been just added to the system.



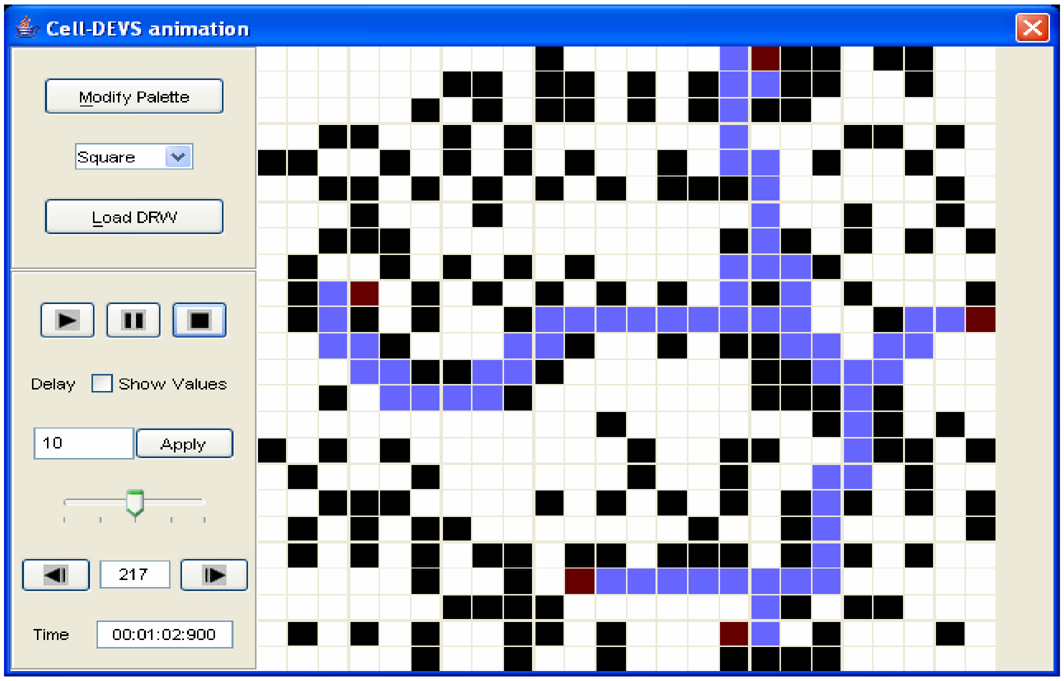
The state of the model after 125 steps of execution is as follows. Here the new node has generated the wave messages and multiple (non-optimal) paths from the trees are being formed.



The state of the model after 150 steps of execution is as follows. Here the new node has found the shortest optimal path and that path is becoming the tree. All other non-optimal paths have been purged and all the clear state nodes have been re-initialized.



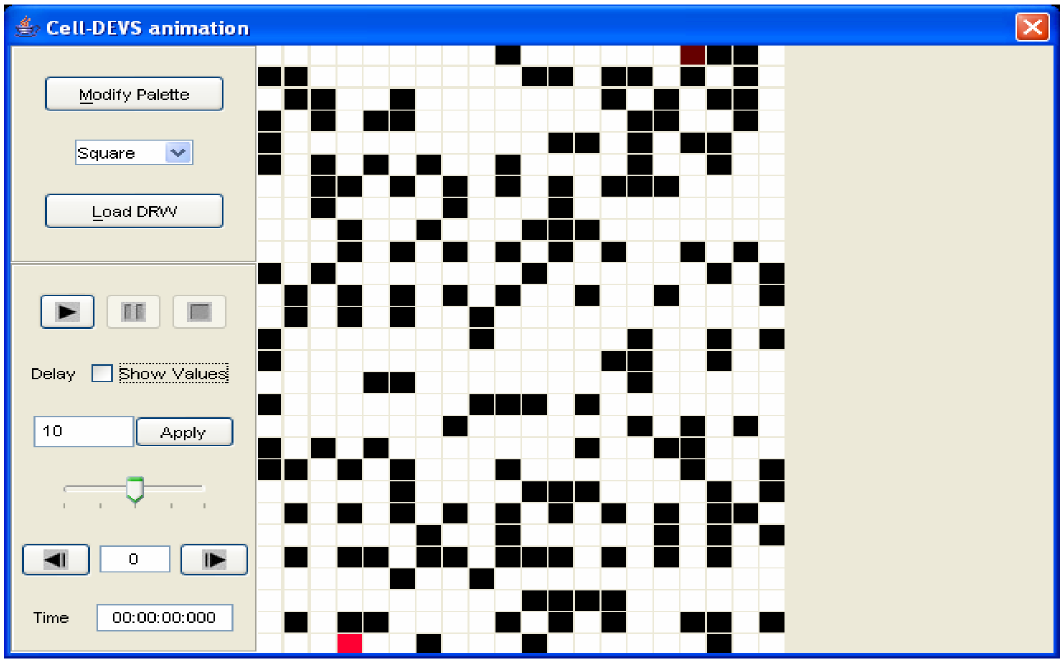
The final state of the model after 217 steps of execution is as follows. Here 3 new nodes have been successfully added to the multicast tree formed between the first two nodes. Note that it is an optimal tree.



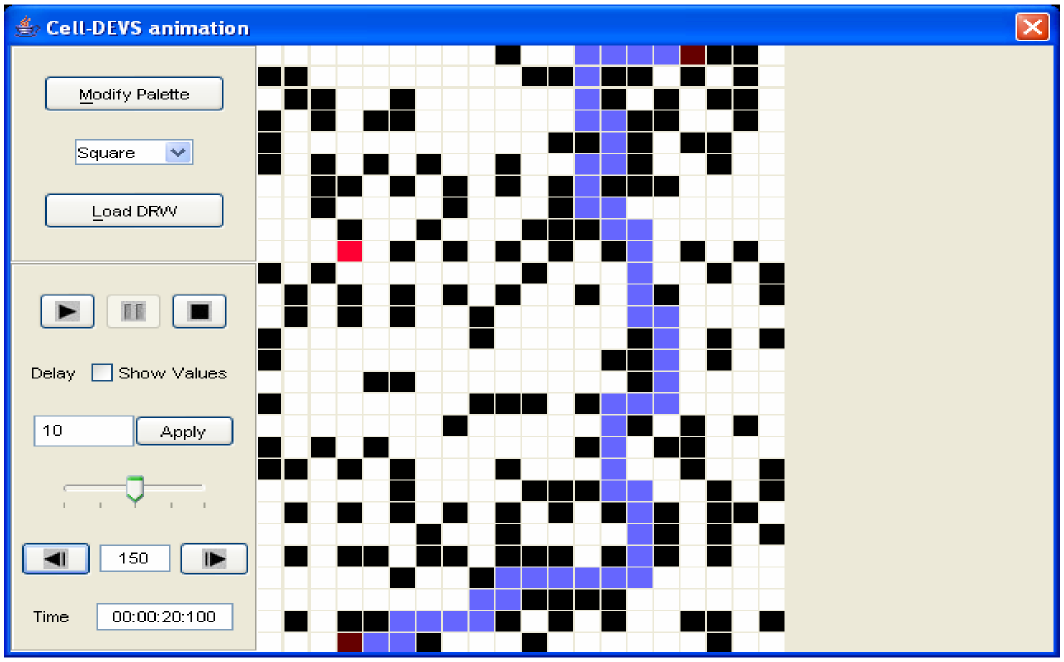
The above test shows that the model has successfully built an optimal multicast tree. Note that during the intermediate steps, many non-optimal paths are generated after the addition of every new node but the model successfully detects and builds the shortest path and purges all other non-optimal paths.

b) Test 2

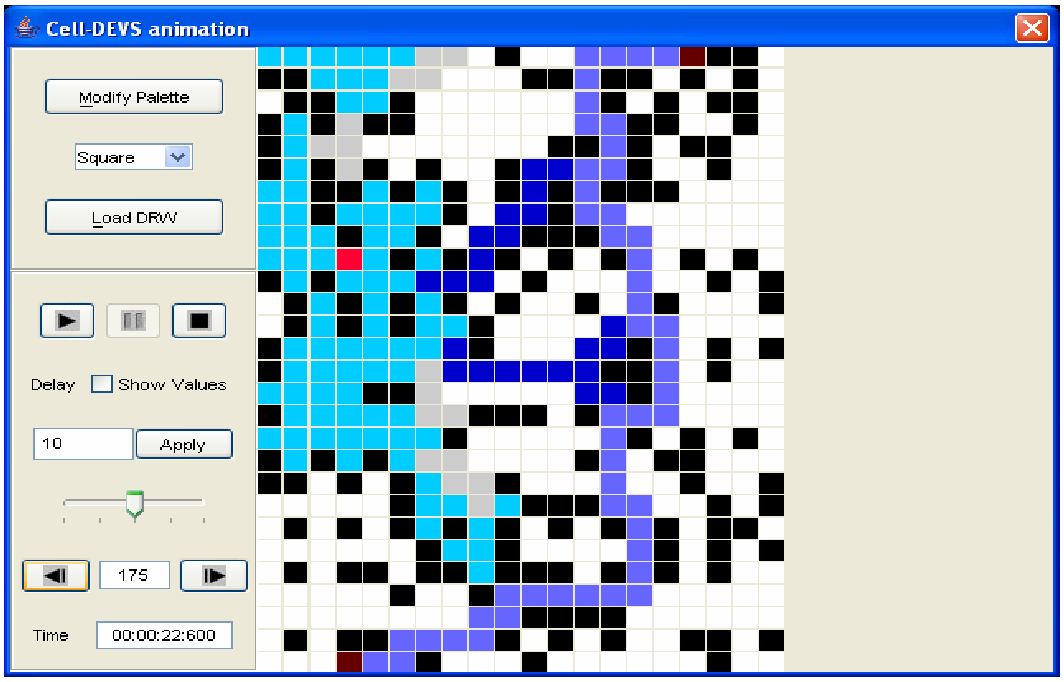
This is a test consisting of 20x28 Cell plane. Initially there are two nodes. Subsequent nodes are added after tree for each of the nodes is completed. The initial distribution of the nodes is given as follows:



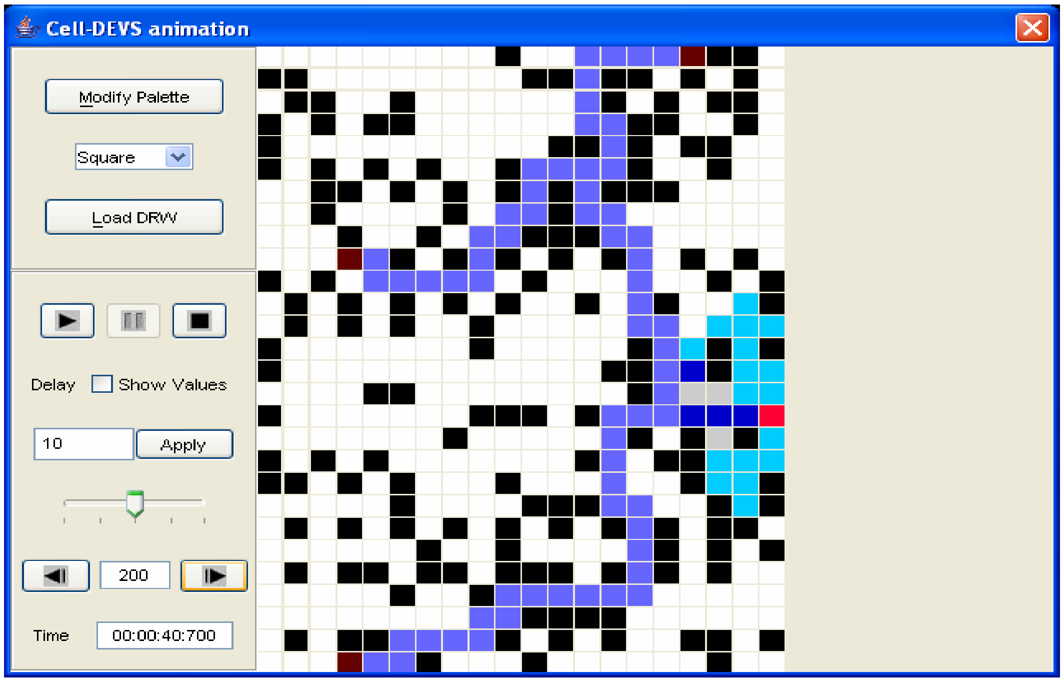
The state of the model after 150 steps of execution is as follows. Here the tree has been established for the first two nodes and an additional node is just added to the system.



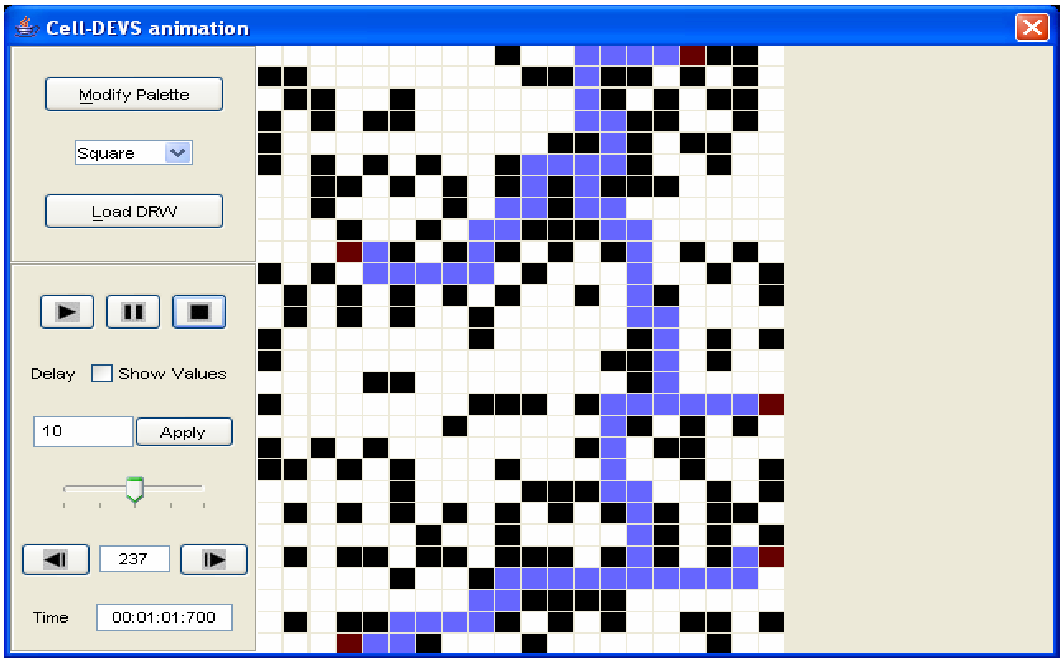
The state of the model after 175 steps of execution is as follows. Here the new node has generated the wave messages and multiple paths from the trees are being formed.



The state of the model after 200 steps of execution is as follows. Here the third node has found the shortest optimal path and that path has become the tree. All other non-optimal paths have been purged and all the clear state nodes have been re-initialized. A fourth node is added to the system which is in the process of finding the optimal path.



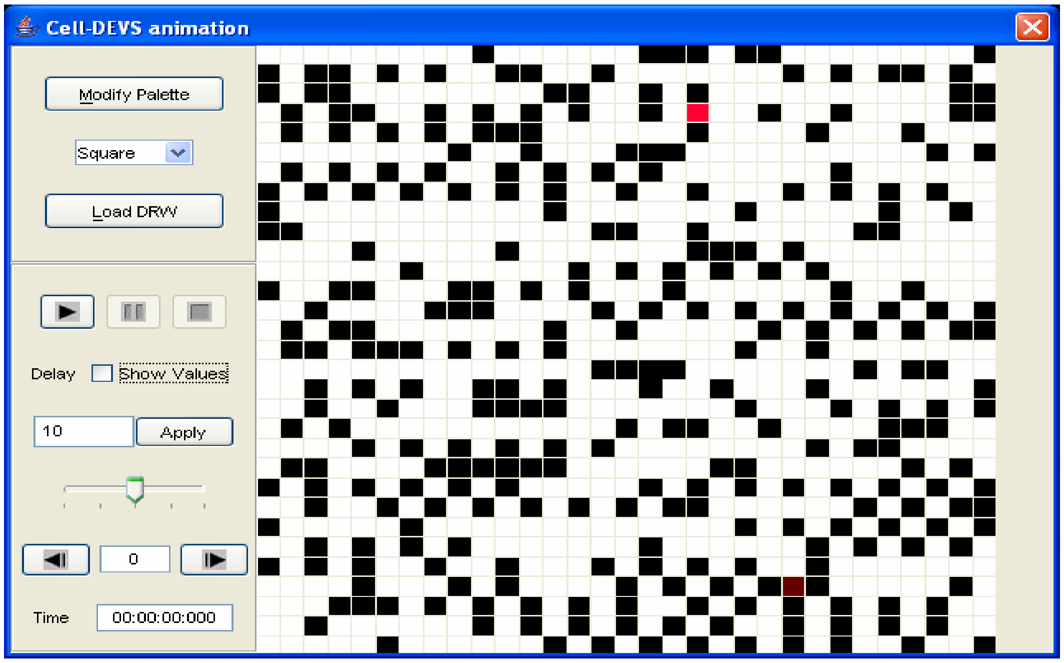
The final state of the model after 237 steps of execution is as follows. Here 3 new nodes have been successfully added to the multicast tree formed between the first two nodes. Note that it is an optimal tree.



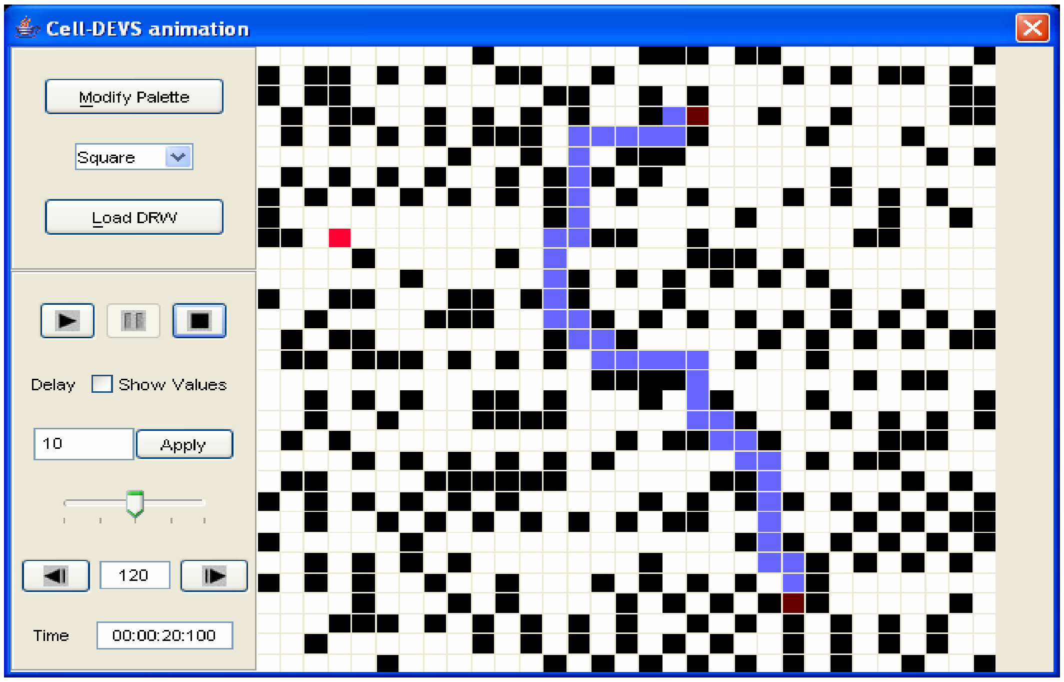
The above test shows that the model has successfully built an optimal multicast tree. Note that during the intermediate steps, many non-optimal paths are generated after the addition of every new node but the model successfully detects and builds the shortest path and purges all other non-optimal paths.

c) Test 3

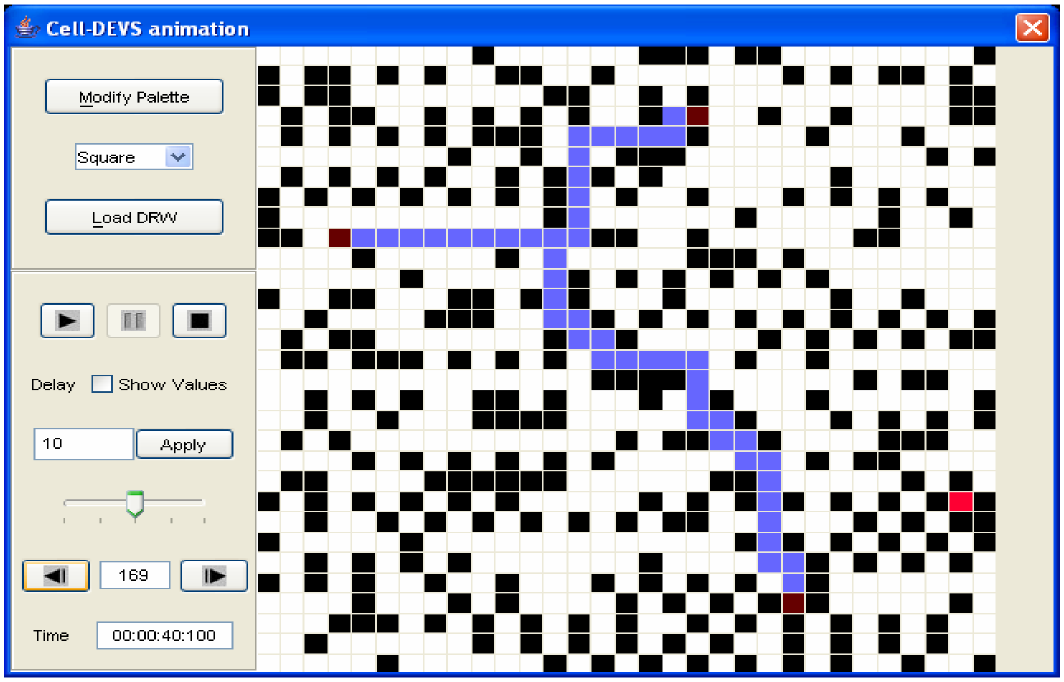
This is a test consisting of 31x31 Cell plane. Initially there are two nodes. Subsequent nodes are added after tree for each of the nodes is completed. The initial distribution of the nodes is given as follows:



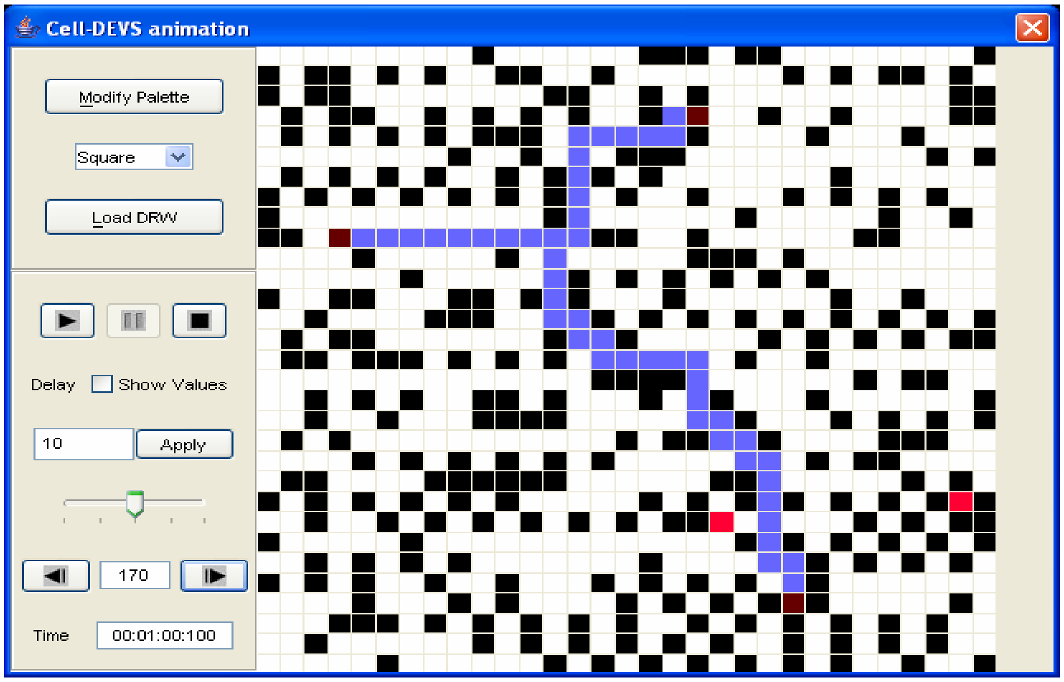
The state of the model after 120 steps of execution is as follows. Here the tree has been established for the first two nodes and an additional node is just added to the system.



The state of the model after 169 steps of execution is as follows. The third node is successfully and optimally has been added to the multicast tree and the fourth node is just added to the system.



The state of the model after 170 steps of execution is as follows. Note that the fourth node added to the system could not join the multicast tree. The reason is that that there does not exist any path from the fourth node to the multicast tree. The fifth node is just added to the system.

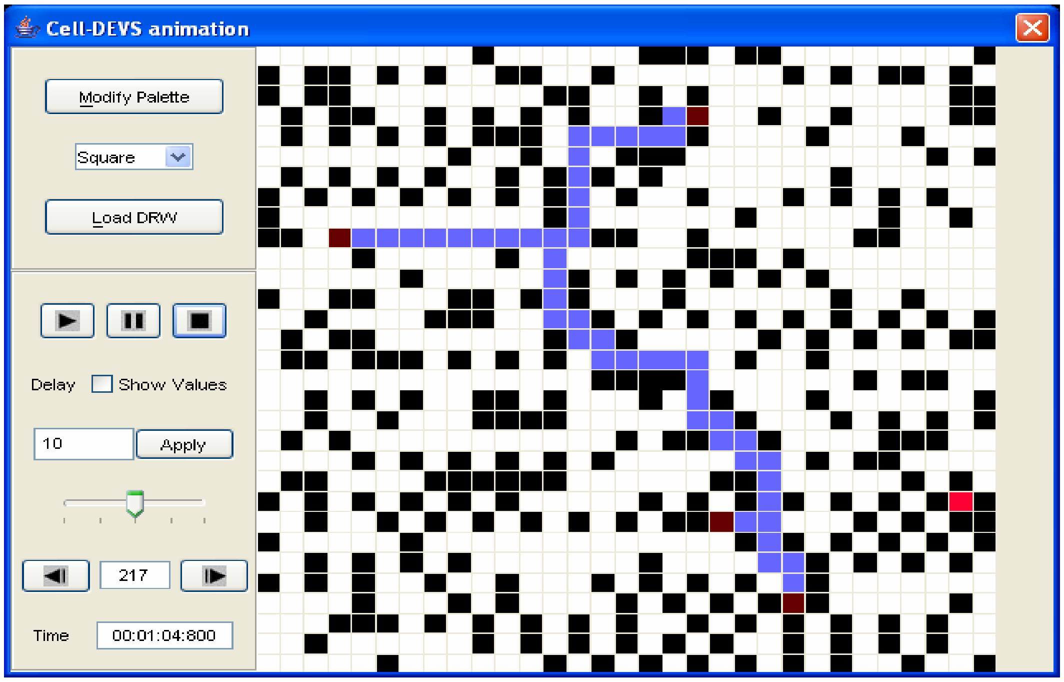


10

170

Time 00:01:00:100

The final state of the model after 217 steps of execution is as follows. Here 2 new nodes have been successfully added to the multicast tree formed between the first two nodes. Note that it is an optimal tree. The fourth node could not be added to the tree because there exists no path from the fourth node to the multicast tree.



10

217

Time 00:01:04:800

The above test shows that the model has successfully built an optimal multicast tree. Note that during the intermediate steps, many non-optimal paths are generated after the addition of every new node but the model successfully detects and builds the shortest path and purges all other non-optimal paths.

3.2.7 Reaction of the Model to Different Inputs than Those Defined in the Specifications

The requirement of the model is that it has one and only one sender and one and only one receiver node in the .map file. The cases that violate this condition have been discussed in detail in Section 3.1.6 and will not be repeated here.

The other requirement of the model is that the new node should not be added to the system until the tree for the previous node has been completed and all wave and clear state nodes have been re-initialized. If this condition is violated, the wave messages for the previous node interfere with the wave messages for the new node. This results in an unexpected and undesired behavior and mostly a successful shortest optimal path to the tree is not made. So the conclusion is that the new node should be added to the system only when the tree for the previous node has been completed and all wave and clear states nodes have been re-initialized.

3.3 Part 3: Routing Among Multiple Pairs of Senders and Receivers

3.3.1 Conceptual Model Description

The conceptual model for this part has been described in Section 1.1 and Section 2.3. It will not be repeated here to save space.

3.3.2 Algorithm Design for Routing Among Multiple Pairs of Senders and Receivers

As discussed in Section 2.3 that the original Lee's Algorithm breaks for multiple pairs of senders and receivers, a new algorithm needs to be devised. The approach taken during this project is that each pair of sender and receiver is allocated one plane in the Cell-DEVS model. The total number of planes in the resulting 3 dimensional model thus depends on the total pair of receivers and senders to be routed. In each plane the original Lee's algorithm is implemented with a total of 15 states for each of the cell. Thus, multiple pairs of senders and receivers and routed, without having to define more states. Moreover, this approach exploits the inherent parallelism in the Cell-DEVS model as many pairs are routed simultaneously without interfering with each other.

3.3.3 Formal Specifications for the Cell-DEVS Model

In order to compute the shortest path between multiple pairs of senders and receivers, a Cell-DEVS model named *path* is defined. Its formal specifications are given below.

CD = $< X, Y, \mathrm{I}, \mathrm{S}, \theta, \mathrm{N}, \mathrm{d}, \delta_{\mathrm{int}}, \delta_{\mathrm{ext}}, \tau, \lambda, \mathrm{D} >$

X = S
Y = S

I = $<\eta, \mu, \mathrm{P}^x, \mathrm{P}^y>$
Where
η = 5
μ = 0
P_j^i = $\{ (\mathrm{N}_j^i, \mathrm{T}_j^i) \,/\, \forall\, j \in [1, 5],\ \mathrm{N}_j^i \in [i_1, i_5]$ and $\mathrm{T}_j^i \in \mathrm{I}_i$ $\}$, $\mathrm{I}_i = \{ x \,/\, x \in X$ if $i = \mathrm{X} \}$ or $\mathrm{I}_i = \{ x \,/\, x \in Y$ if $i = \mathrm{Y} \}$;

S = $\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14\}$

θ = { (s, phase, σqueue, σ) /
s $\in$ S is the state value for a cell (S has already been defined),
phase $\in$ {active, passive},
σqueue = $\{ ((v_1,\sigma_1),\ldots,(v_m,\sigma_m)) \,/\, m \in N \wedge m < \infty) \wedge \forall\, (i \in N, i \in [1,m]), v_i \in \mathrm{S} \wedge \sigma_i \in R_0^{+} \cup \infty\}$; and
$\sigma \in R_0^{+} \cup \infty$
} (As Transport delay is used)

N = $\{(-1,0,0), (0,-1,0), (0,0,0), (0,1,0), (1,0,0)\}$

d = 100

D = $\theta \times \mathrm{N} \times \mathrm{d} \rightarrow R_0^{+} \cup \infty$,

For δ_{int}, δ_{ext}, λ and τ see path.ma (These functions actually constitute the whole algorithm that has been given in Section 3.3.2 and implemented in path.ma. These functions are hence not written here to save space).

GCC = $<$ Xlist, Ylist, I, X, Y, η, N, {f, c, b}, C, B, Z, select $>$

Xlist = { Ø }
Ylist = { Ø }

I = $< \mathrm{P}^x, \mathrm{P}^y >$
Where
P^x = { Ø }
P^y = { Ø }

X = S (Defined Above)

Y	=	S (Defined Above)
η	=	5
N	=	{(-1,0,0), (0,-1,0), (0,0,0), (0,1,0), (1,0,0)}
f	=	Different values are used for different tests e.g. 5, 25, 30 etc.
c	=	Different values are used for different tests e.g. 5, 25, 30 etc.
b	=	2
C	=	$\{C_{ijk} / i \in [1,f] \wedge j \in [1,c] \wedge k \in [1,b]\}$
B	=	No-wrapped

Z =

$P_{ij}\, Y_1 \rightarrow P_{i,j-1}\, X_1$ $\qquad$ $P_{i,j+1}\, Y_1 \rightarrow P_{ij}\, X_1$

$P_{ij}\, Y_2 \rightarrow P_{i+1,j}\, X_2$ $\qquad$ $P_{i-1,j}\, Y_2 \rightarrow P_{ij}\, X_2$

$P_{ij}\, Y_3 \rightarrow P_{i,j+1}\, X_3$ $\qquad$ $P_{i,j-1}\, Y_3 \rightarrow P_{ij}\, X_3$

$P_{ij}\, Y_4 \rightarrow P_{i-1,j}\, X_4$ $\qquad$ $P_{i+1,j}\, Y_4 \rightarrow P_{ij}\, X_4$

$P_{ij}\, Y_5 \rightarrow P_{ij}\, X_5$ $\qquad$ $P_{ij}\, Y_5 \rightarrow P_{ij}\, X_5$

Select = { (-1,0,0), (1,0,0), (0,0,0), (0,1,0), (0,-1,0)}

3.3.4 Test Strategy

The model will be tested by having different initial distributions of the node in each plane and checking whether the algorithm successfully determines the shortest path between two nodes in each plane (if one exists). The initial distribution of the nodes would be provided as an input to the model in terms of .map files. The .map file provided as an input to the model should have the following characteristics.

- There is only one start point in each plane.
- There is only one end point in each plane.
- There is at least one start point in each plane.
- There is at least one end point in each plane.

The first two conditions are necessary because the Lee's algorithm fails if there are multiple senders or receivers in the same plane. The last two points are the requirements of the problem statement. Note that there may no path exists between the starting and the end point. In such a situation, the system would end up with some nodes in the wave conditions and the other nodes in initial state. If more than one path exists, the system would locate the shortest path.

3.3.5 Implementation in CD++

For implementation of the algorithm in CD++, a total of 15 states have been defined. These 15 states along with their state values are given as follows.

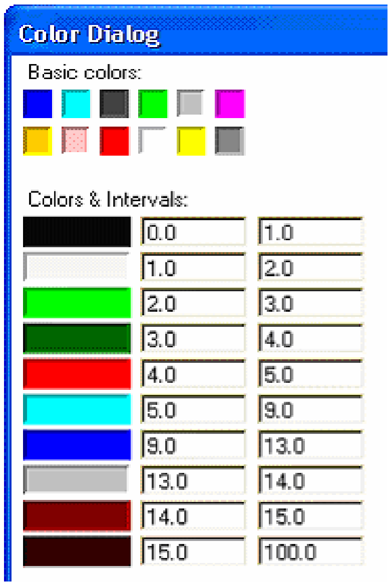
State 0	Dead Cell (Broken Communication Link)
State 1	Initial State of the Nodes
State 2	Initial State of the Destination Node
State 3	Destination Ready (State of the Destination Node after it has received a send request from the sender)
State 4	Initial State of the Sender Node
State 5	Wave Up ↑
State 6	Wave Down ↓
State 7	Wave Right →
State 8	Wave Left ←
State 9	Path Up ⇑
State 10	Path Down ⇓
State 11	Path Right ⇒
State 12	Path Left ⇐
State 13	Clear State (Intermediate State of the nodes that received a wave message but are not going to become the path)
State 14	Destination Found (Final State of the Sender Node)

Using these state values the model was implemented in CD++. The path.ma file is presented next. Note here that in [path-rule] many Boolean statements could have been combined together. However, for the sake of clarity each statement is written in a separate line. The *initialMapValue* has been defined as path.map. This is the name of the file that the model takes as an input. Transport delay is used and a no-wrapped border is used. The dimensions of the model given here are for a particular example. Different tests were run with different values of width and height.

```
[top]
components : path

[path]
type : cell
```

```
dim : (2,5,5)
delay : transport
defaultDelayTime : 100
border : nowrapped
neighbors :             path(0,-1,0)
neighbors : path(0,0,-1)  path(0,0,0)  path(0,0,1)
neighbors :             path(0,1,0)
initialvalue : 1
initialMapValue : path.map
localtransition : path-rule

[path-rule]
rule : 3 100 { (0,0,0) = 2 and stateCount(9) > 0}
rule : 3 100 { (0,0,0) = 2 and stateCount(10) > 0}
rule : 3 100 { (0,0,0) = 2 and stateCount(11) > 0}
rule : 3 100 { (0,0,0) = 2 and stateCount(12) > 0}

rule : 5 100 { (0,0,0) = 1 and (0,-1,0) > 3 and (0,-1,0) < 9}
rule : 6 100 { (0,0,0) = 1 and (0,1,0) > 3 and (0,1,0) < 9}
rule : 7 100 { (0,0,0) = 1 and (0,0,1) > 3 and (0,0,1) < 9}
rule : 8 100 { (0,0,0) = 1 and (0,0,-1) > 3 and (0,0,-1) < 9}

rule : 9 100 { (0,0,0) = 5 and stateCount(2) = 1}
rule : 10 100 { (0,0,0) = 6 and stateCount(2) = 1}
rule : 11 100 { (0,0,0) = 7 and stateCount(2) = 1}
rule : 12 100 { (0,0,0) = 8 and stateCount(2) = 1}
rule : 9 100 { (0,0,0) = 5 and (0,0,-1) = 11}
rule : 9 100 { (0,0,0) = 5 and (0,0,1) = 12}
rule : 9 100 { (0,0,0) = 5 and (0,1,0) = 9}
rule : 10 100 { (0,0,0) = 6 and (0,0,-1) = 11}
rule : 10 100 { (0,0,0) = 6 and (0,0,1) = 12}
rule : 10 100 { (0,0,0) = 6 and (0,-1,0) = 10}
rule : 11 100 { (0,0,0) = 7 and (0,0,-1) = 11}
rule : 11 100 { (0,0,0) = 7 and (0,-1,0) = 10}
rule : 11 100 { (0,0,0) = 7 and (0,1,0) = 9}
rule : 12 100 { (0,0,0) = 8 and (0,0,1) = 12}
rule : 12 100 { (0,0,0) = 8 and (0,-1,0) = 10}
rule : 12 100 { (0,0,0) = 8 and (0,1,0) = 9}

rule : 13 100 { (0,0,0) = 1 and stateCount(13) > 0}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and stateCount(13) > 0}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and stateCount(3) > 0}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and stateCount(14) > 0}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and (0,-1,0) > 8 and
       (0,-1,0) < 13 and (0,-1,0) != 10}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and (0,1,0) > 8 and (0,1,0)
      < 13 and (0,1,0) != 9}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and (0,0,-1) > 8 and
       (0,0,-1) < 13 and (0,0,-1) != 11}
rule : 13 100 { (0,0,0) > 4 and (0,0,0) < 9 and (0,0,1) > 8 and (0,0,1)
      < 13 and (0,0,1) != 12}

rule : 14 100 { (0,0,0) = 4 and stateCount(9) > 0}
rule : 14 100 { (0,0,0) = 4 and stateCount(10) > 0}
rule : 14 100 { (0,0,0) = 4 and stateCount(11) > 0}
rule : 14 100 { (0,0,0) = 4 and stateCount(12) > 0}
rule : {(0,0,0)} 100 {t}
```

In order to for the results to be viewed in VRML GUI a color palette is defined. This palette is presented next as it is necessary to explain the results presented in the next section. Note here that in the palette all the *wave* states of the nodes (state 5 to state 8) are represented with the same color. Similarly, all the *path* states of the node (state 9 to state 12) are being represented by the same color.



⬛	0.0	1.0
⬜	1.0	2.0
🟩	2.0	3.0
🟩	3.0	4.0
🟥	4.0	5.0
🟦	5.0	9.0
🟦	9.0	13.0
▨	13.0	14.0
🟥	14.0	15.0
⬛	15.0	100.0

Figure 3.3: Color Palette for Viewing Results of Part 3 in VRML GUI

Thus all the dead cells are represented as black cells and all the cells that have not received any message yet represented as white. The destination is represented as light green but after receiving the send request from the sender it changes its color to dark green. Sender is red. All the wave messages are light blue and the path nodes dark blue. Those wave nodes that are not going to become the path gets to clear state which is represented as grey. The sender after successful discovery of the path to the destination changes its color to maroon.

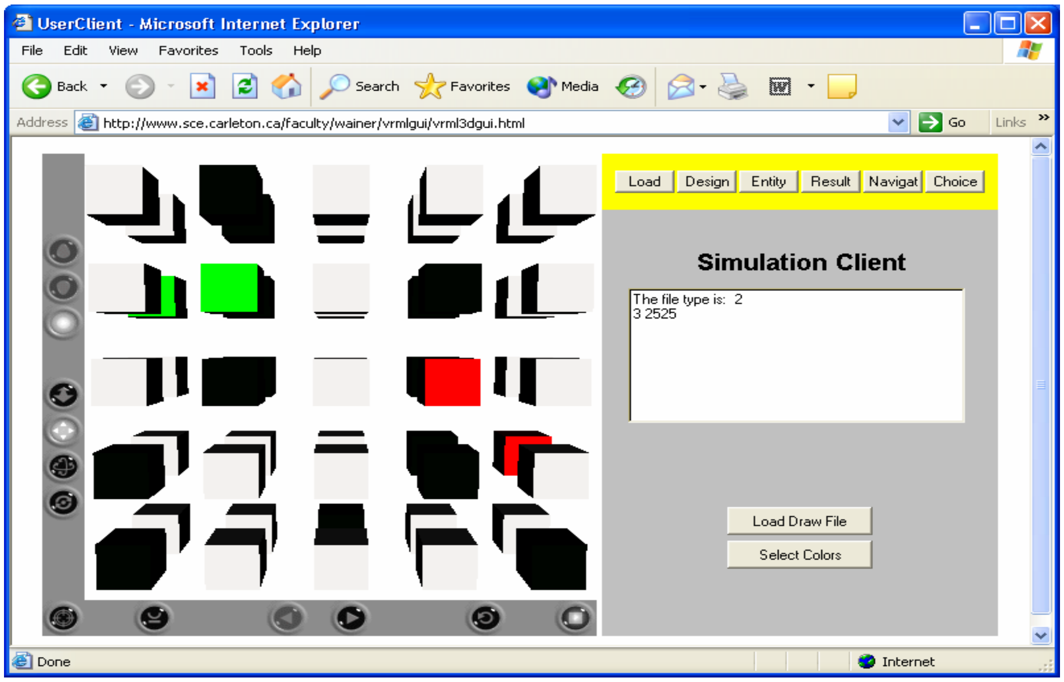
3.3.6 Testing

A number of tests were conducted on the model. A subset of these tests is presented in this section. Note that these tests are given in the attached diskette. As the input of each test is a map file which is very long, it is not included in the report. Similarly, .drw files are very long, so they are not included in the report either. The screen shots taken from execution of the model on the Modeler at different intervals during the test are given in this report.

Due to the visualization difficulties of 3 dimensional models, the tests given below have 2 planes only and thus they route two pairs of senders and receivers, one in each plane. However, the algorithm and model discussed above is equally valid for higher number of planes. It has been tested but the results are not given here because of visual limitations of such tests.

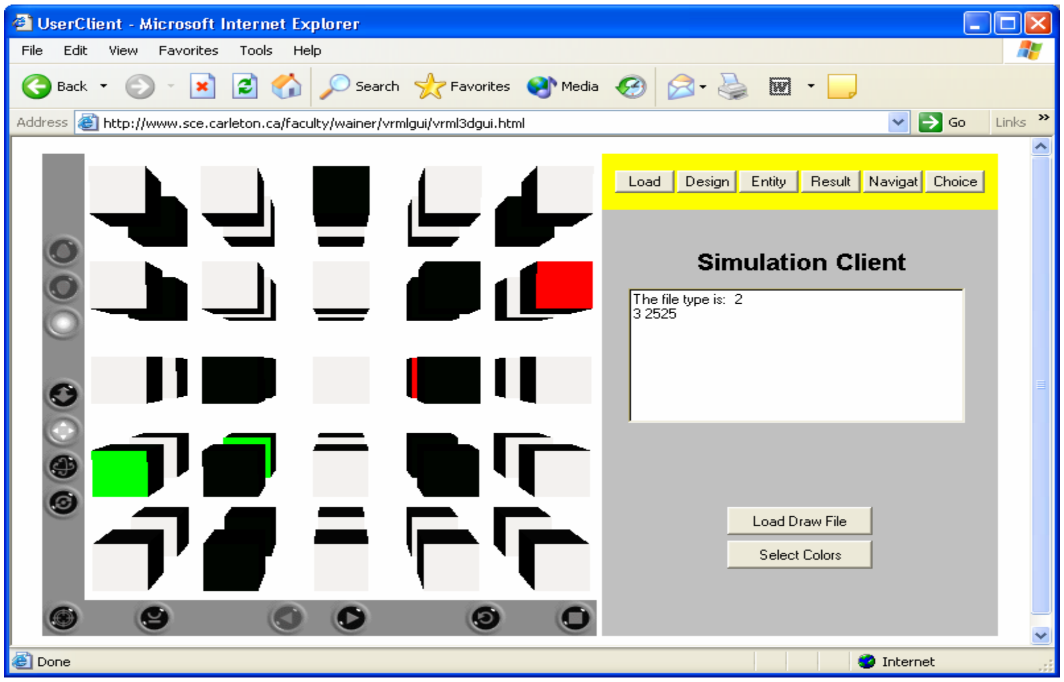
a) Test 1

This is a simple test consisting of 5x5 cells space having 2 planes and thus route two pairs of senders and receivers, one in each plane. The initial distribution of nodes in plane 1 is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Simulation Client
Load Draw File
Select Colors
Done
Internet

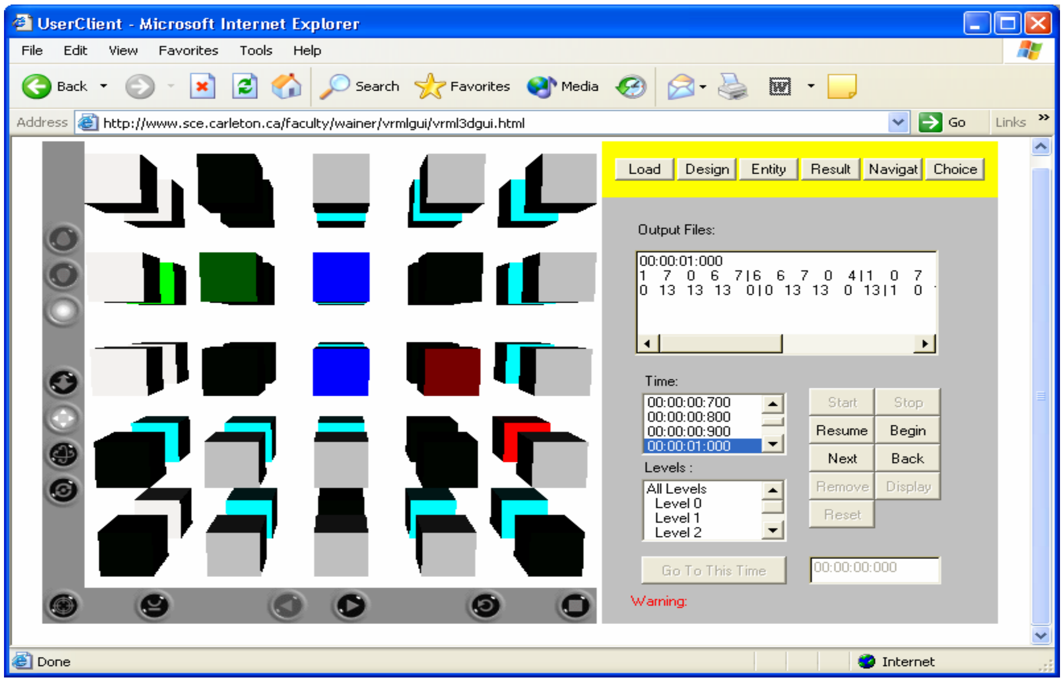
The file type is: 2
3 2525

The initial distribution of nodes in plane 2 is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Simulation Client
Load Draw File
Select Colors
Done
Internet

The file type is: 2
3 2525

The state of the nodes in plane 1 after 10 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Done
Internet

Output Files:

```
00:00:01:000
1   7   0   6   7|6   6   7   0   4|1   0   7
0  13  13  13   0|0  13  13   0  13|1   0
```

Time:

```
00:00:00:700
00:00:00:800
00:00:00:900
00:00:01:000
```

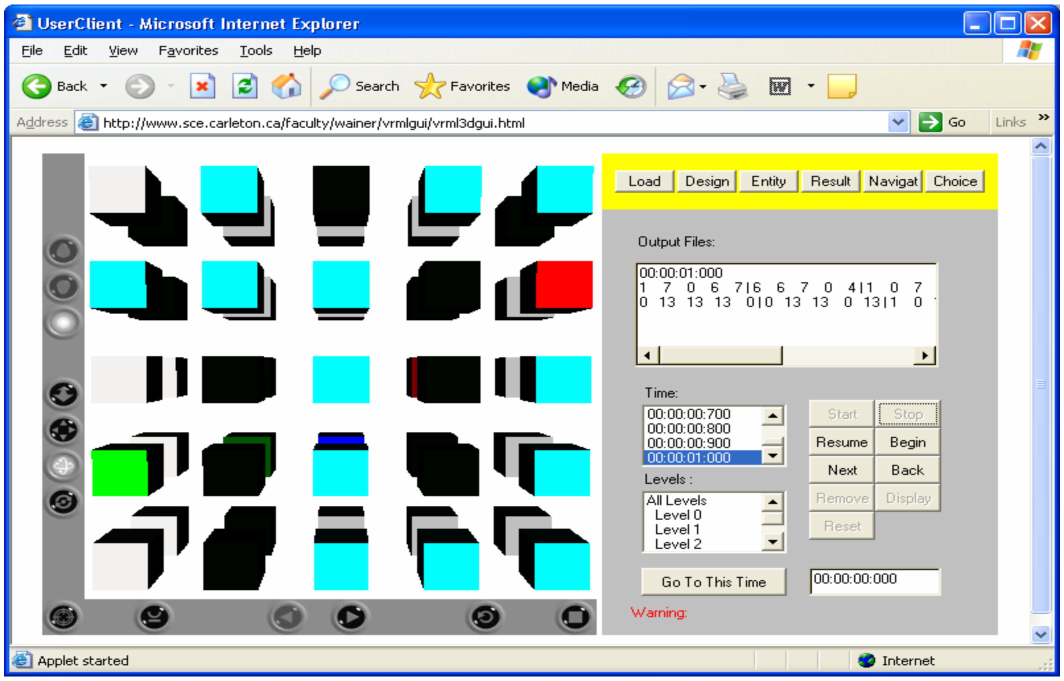
Levels :

```
All Levels
  Level 0
  Level 1
  Level 2
```

00:00:00:000

Warning:

The state of the nodes in plane 2 after 10 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Applet started
Internet

Output Files:

```
00:00:01:000
1   7   0   6   7|6   6   7   0   4|1   0   7
0  13  13  13   0|0  13  13   0  13|1   0
```

Time:

```
00:00:00:700
00:00:00:800
00:00:00:900
00:00:01:000
```

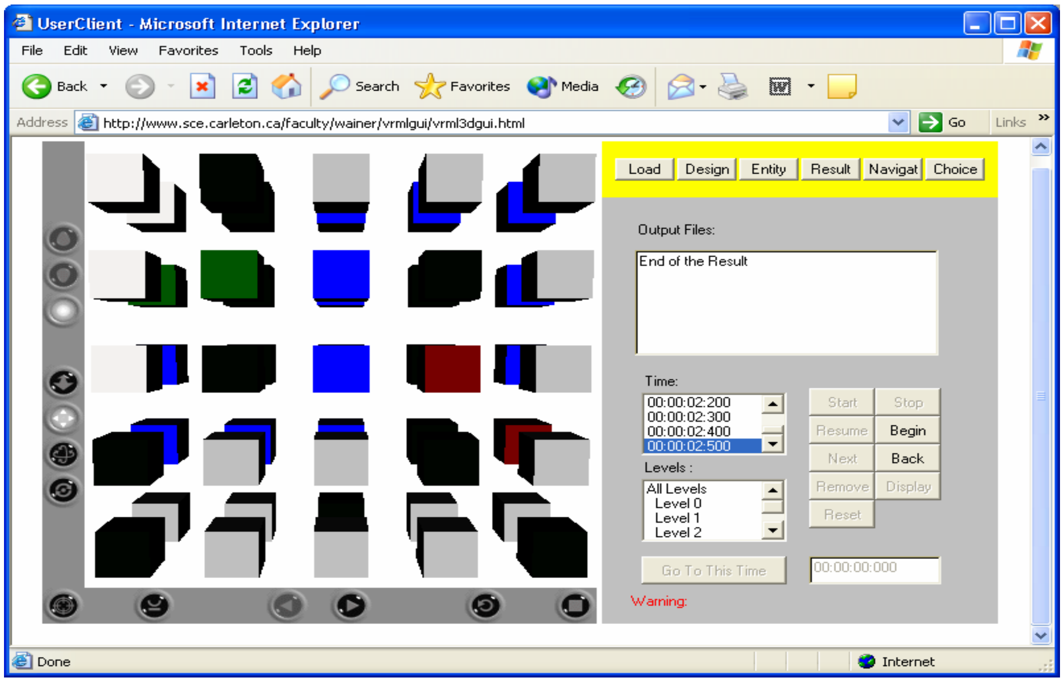
Levels :

```
All Levels
  Level 0
  Level 1
  Level 2
```

00:00:00:000

Warning:

The final state of the nodes in plane 1 after 25 steps of execution is as follows:



UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Done
Internet

Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

Output Files:

```
End of the Result
```

Time:

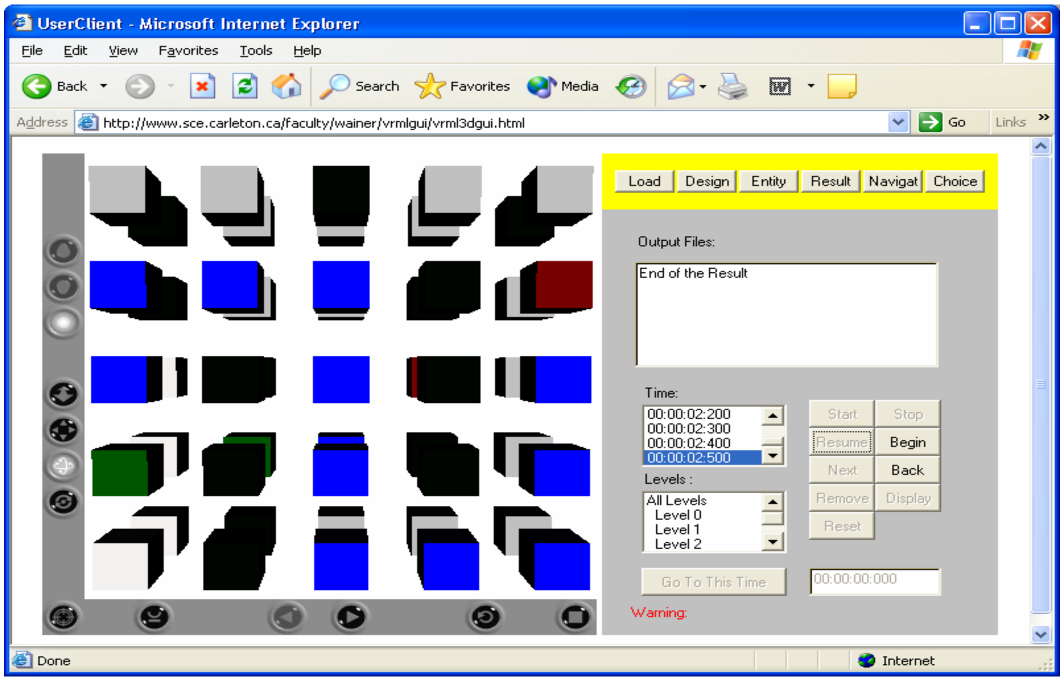
```
00:00:02:200
00:00:02:300
00:00:02:400
00:00:02:500
```

Levels :

```
All Levels
  Level 0
  Level 1
  Level 2
```

00:00:00:000

The final state of the nodes in plane 2 after 25 steps of execution is as follows:



UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Done
Internet

Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

Output Files:

```
End of the Result
```

Time:

```
00:00:02:200
00:00:02:300
00:00:02:400
00:00:02:500
```

Levels :

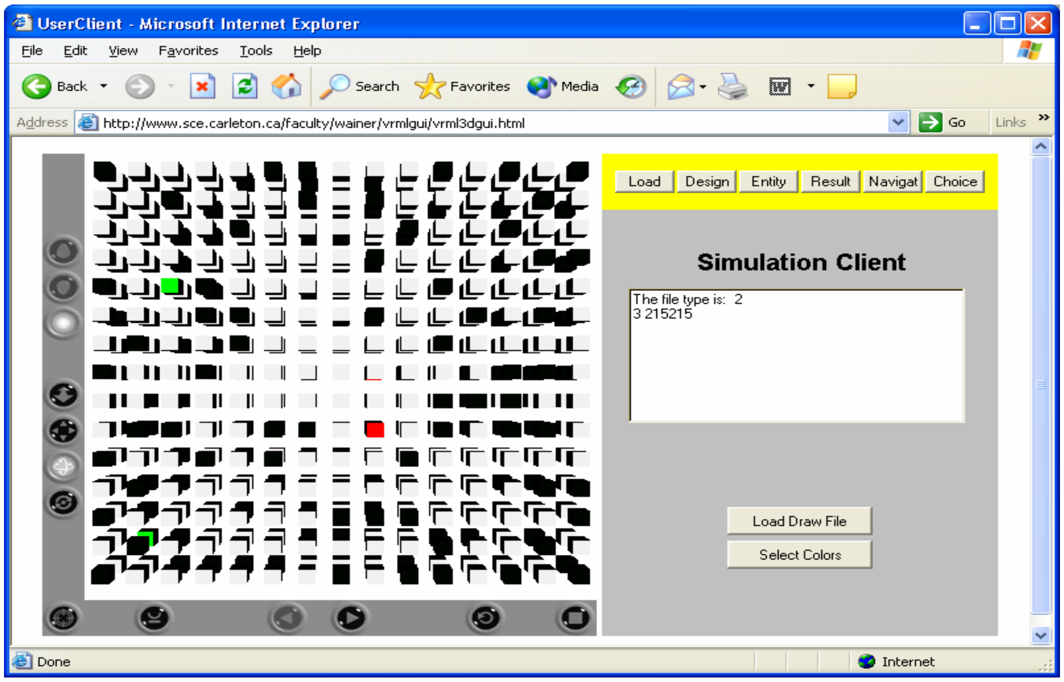
```
All Levels
  Level 0
  Level 1
  Level 2
```

00:00:00:000

The results thus show that the model has successfully established the shortest path between the sender and the receiver in each of the planes. The algorithm thus is capable of routing among multiple pairs of senders and receivers simultaneously without having to define more states.

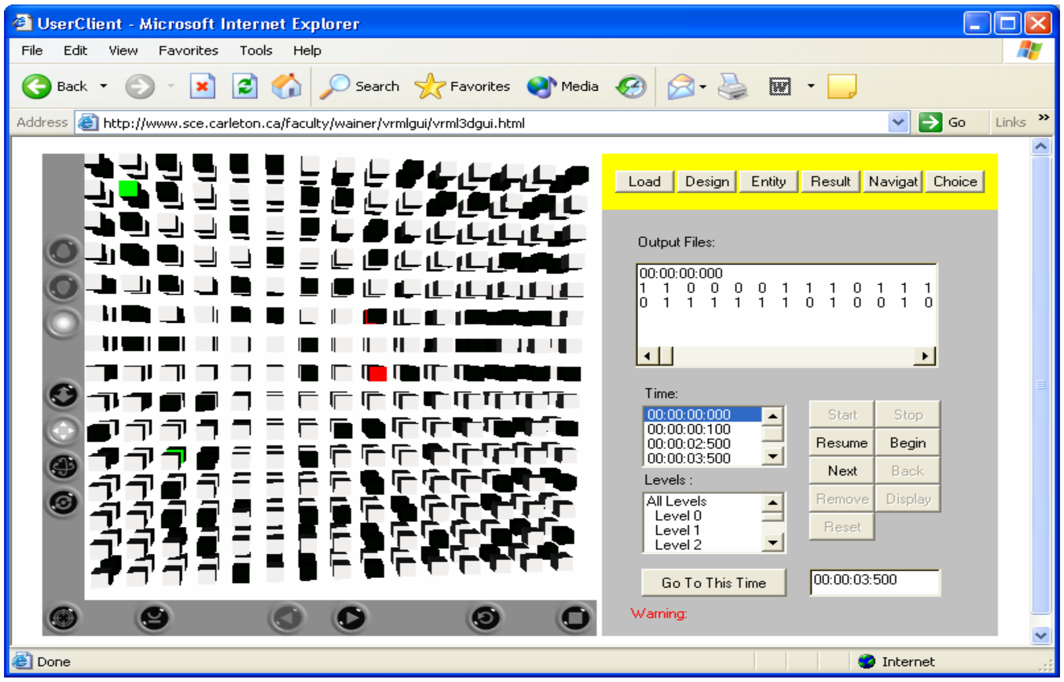
b) Test 2

This is a simple test consisting of 15x15 cells space having 2 planes and thus route two pairs of senders and receivers, one in each plane. The initial distribution of nodes in plane 1 is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Simulation Client
Load Draw File
Select Colors
Done
Internet

```
The file type is:  2
3 215215
```

The initial distribution of nodes in plane 2 is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Done
Internet

Output Files:

```
00:00:00:000
1   1   0   0   0   0   1   1   1   0   1   1   1
0   1   1   1   1   1   1   0   1   0   0   1   0
```

Time:

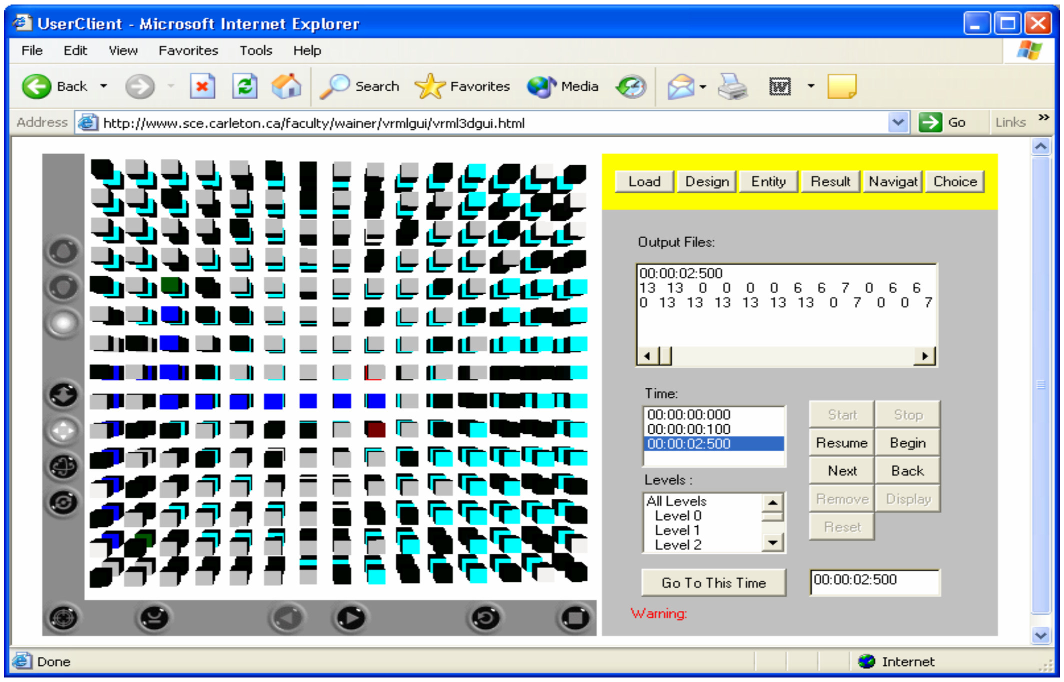
```
00:00:00:000
00:00:00:100
00:00:02:500
00:00:03:500
```

Levels :

```
All Levels
  Level 0
  Level 1
  Level 2
```

00:00:03:500

The state of the nodes in plane 1 after 25 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Done Internet

Output Files:

```
00:00:02:500
13  13   0   0   0   0   6   6   7   0   6   6
0  13  13  13  13  13  13   0   7   0   0   7
```

Time:

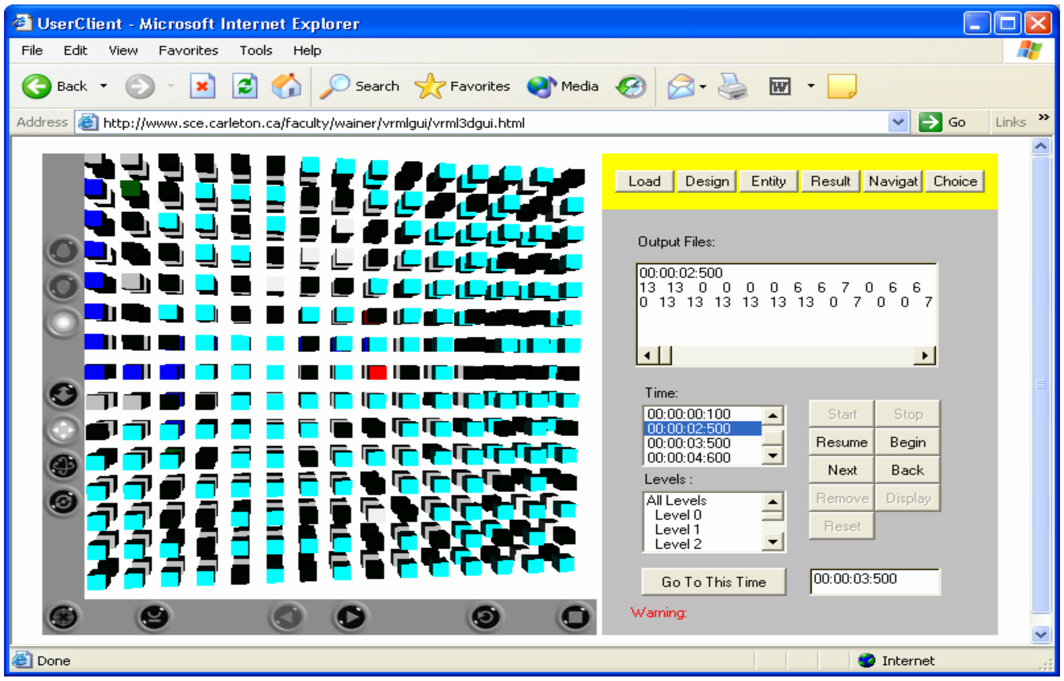
```
00:00:00:000
00:00:00:100
00:00:02:500
```

Levels:

```
All Levels
Level 0
Level 1
Level 2
```

00:00:02:500

The state of the nodes in plane 2 after 25 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Done Internet

Output Files:

```
00:00:02:500
13  13   0   0   0   0   6   6   7   0   6   6
0  13  13  13  13  13  13   0   7   0   0   7
```

Time:

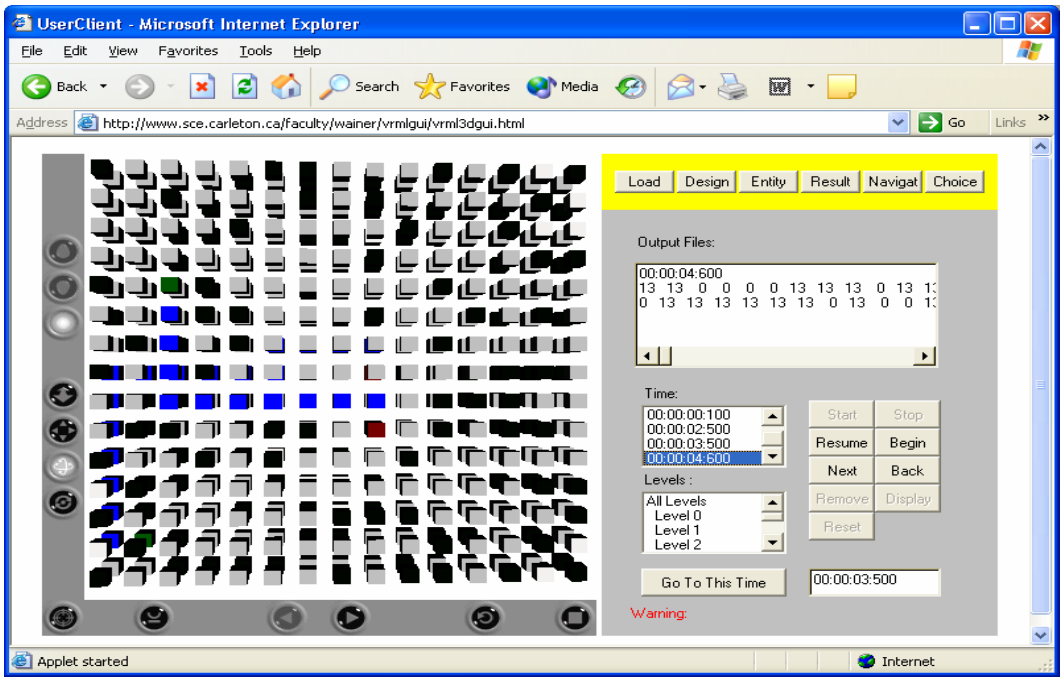
```
00:00:00:100
00:00:02:500
00:00:03:500
00:00:04:600
```

Levels:

```
All Levels
Level 0
Level 1
Level 2
```

00:00:03:500

The final state of the nodes in plane 1 after 46 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Applet started
Internet

Output Files:

```
00:00:04:600
13  13   0   0   0   0  13  13  13   0  13  1
0  13  13  13  13  13  13   0  13   0   0  1
```

Time:

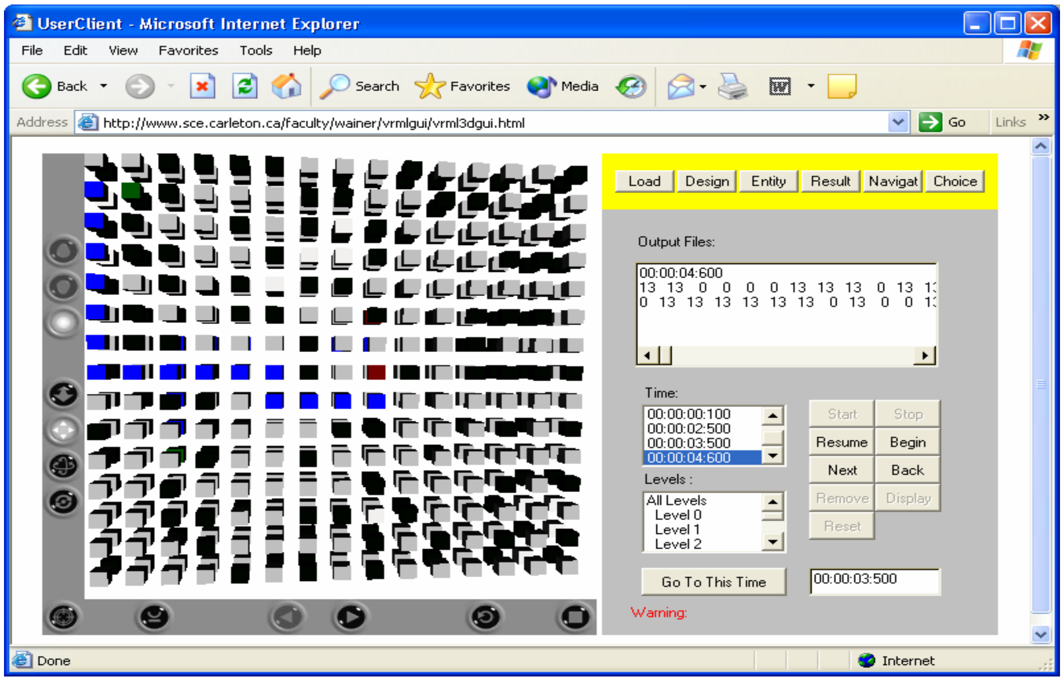
```
00:00:00:100
00:00:02:500
00:00:03:500
00:00:04:600
```

Levels :

```
All Levels
 Level 0
 Level 1
 Level 2
```

```
00:00:03:500
```

The final state of the nodes in plane 2 after 46 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Done
Internet

Output Files:

```
00:00:04:600
13  13   0   0   0   0  13  13  13   0  13  1
0  13  13  13  13  13  13   0  13   0   0  1
```

Time:

```
00:00:00:100
00:00:02:500
00:00:03:500
00:00:04:600
```

Levels :

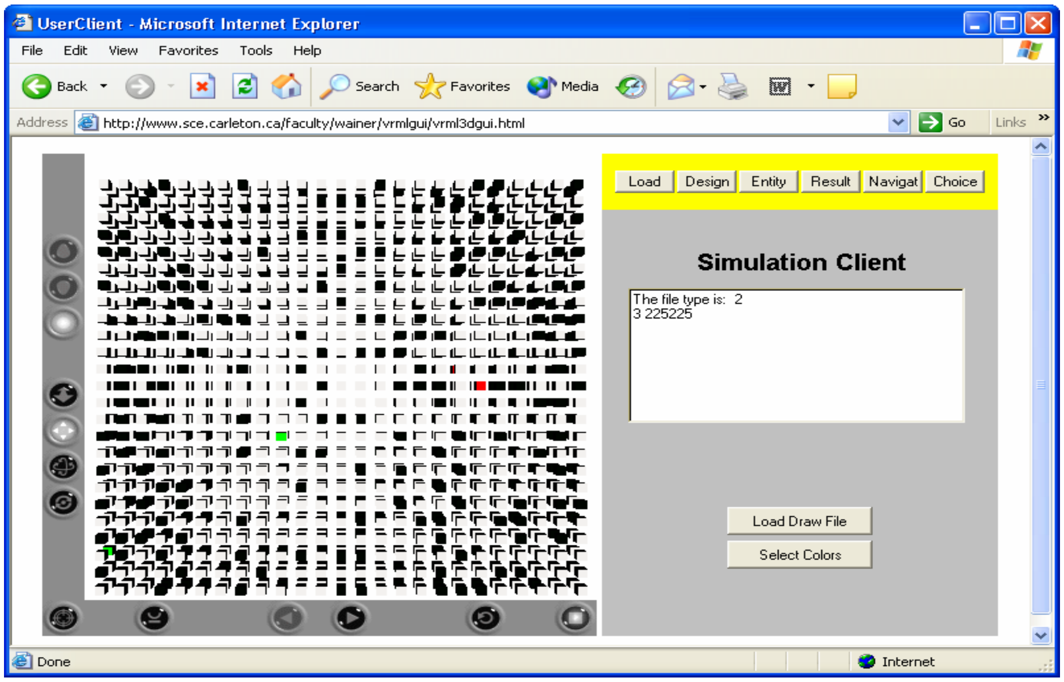
```
All Levels
 Level 0
 Level 1
 Level 2
```

```
00:00:03:500
```

The results thus show that the model has successfully established the shortest path between the sender and the receiver in each of the planes. The algorithm thus is capable of routing among multiple pairs of senders and receivers simultaneously without having to define more states.

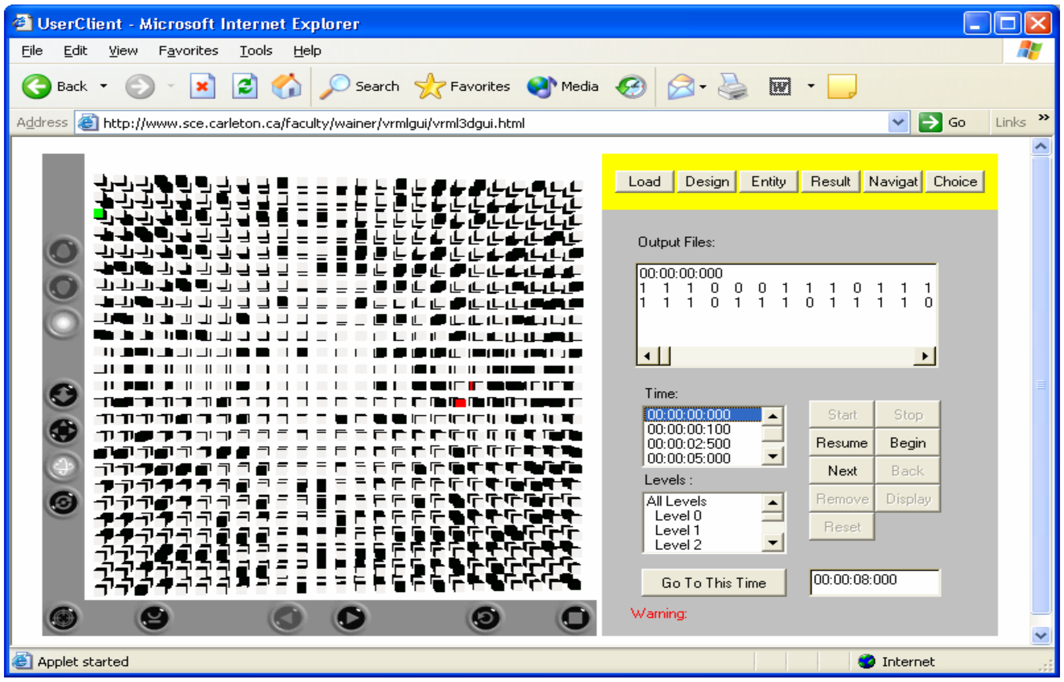
c) Test 3

This is a simple test consisting of 25x25 cells space having 2 planes and thus route two pairs of senders and receivers, one in each plane. The initial distribution of nodes in plane 1 is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Simulation Client
Load Draw File
Select Colors
Done
Internet

```
The file type is:  2
3 225225
```

The initial distribution of nodes in plane 2 is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Warning:
Applet started
Internet

Output Files:

```
00:00:00:000
1   1   1   0   0   0   1   1   1   0   1   1   1
1   1   1   0   1   1   1   0   1   1   1   1   0
```

Time:

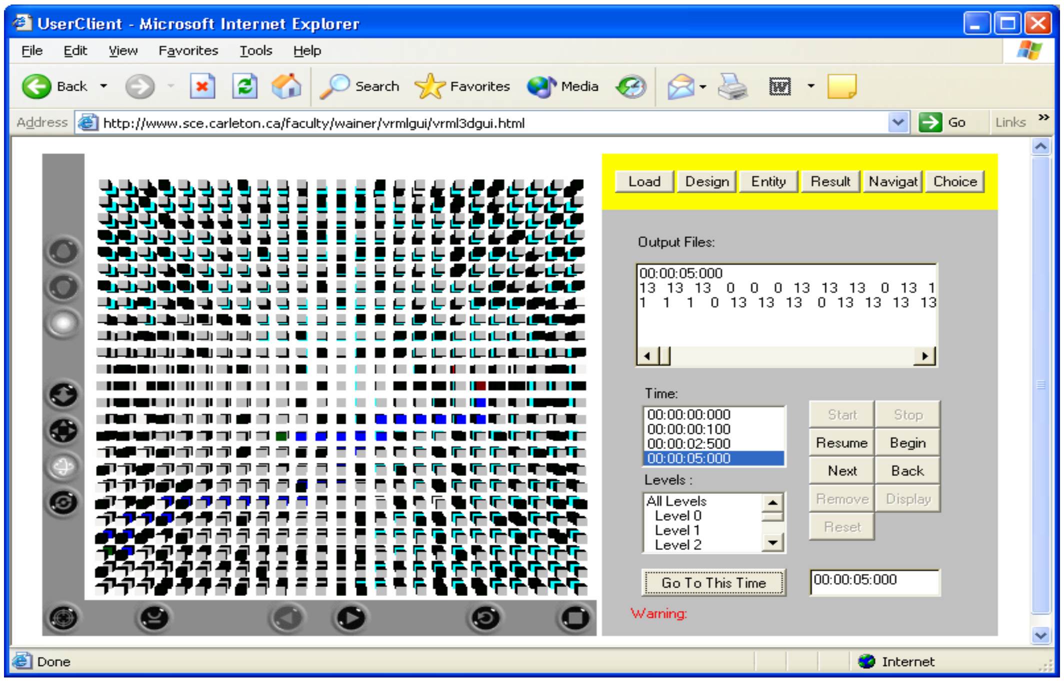
```
00:00:00:000
00:00:00:100
00:00:02:500
00:00:05:000
```

Levels :

```
All Levels
  Level 0
  Level 1
  Level 2
```

00:00:08:000

The state of the nodes in plane 1 after 50 steps of execution is as follows:

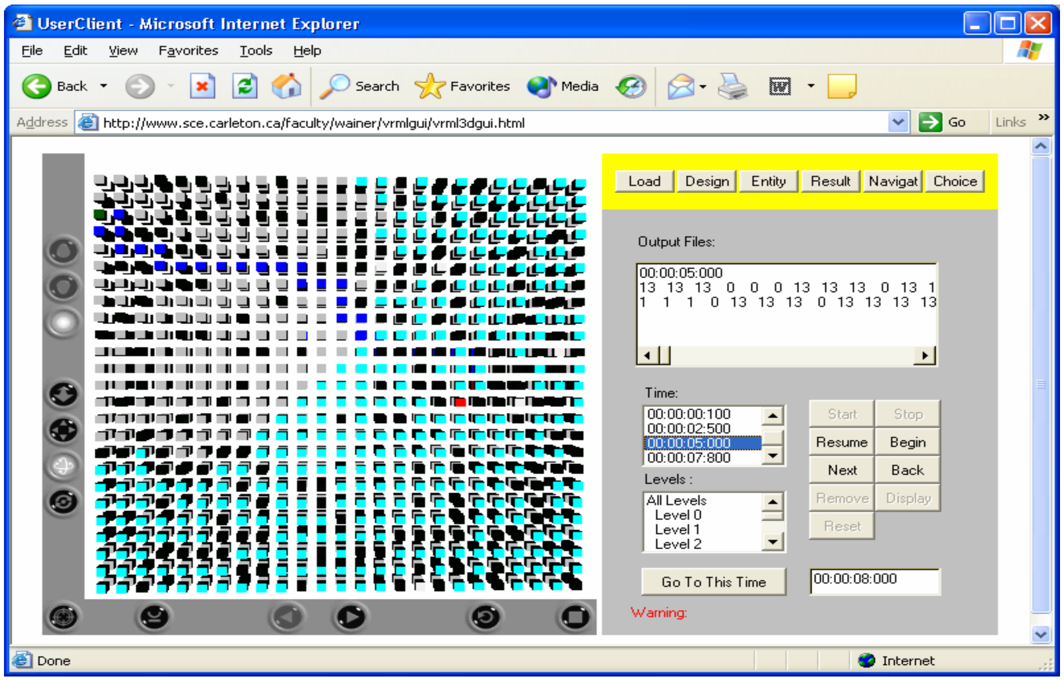

UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Output Files:
Time:
Start Stop Resume Begin Next Back Remove Display Reset
Levels :
All Levels
Level 0
Level 1
Level 2
Go To This Time
Warning:
Done
Internet

```
00:00:05:000
13  13  13   0   0   0  13  13  13   0  13  1
1   1   1   0  13  13  13   0  13  13  13  13
```

```
00:00:00:000
00:00:00:100
00:00:02:500
00:00:05:000
```

00:00:05:000

The state of the nodes in plane 2 after 50 steps of execution is as follows:

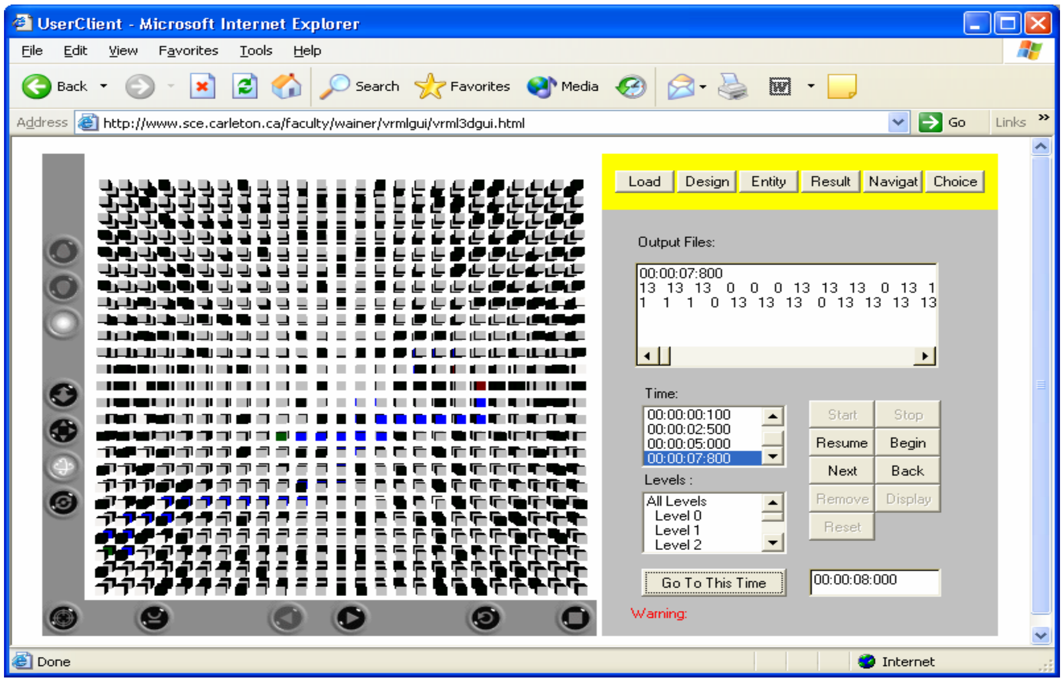

UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html
Go Links
Load Design Entity Result Navigat Choice
Output Files:
Time:
Start Stop Resume Begin Next Back Remove Display Reset
Levels :
All Levels
Level 0
Level 1
Level 2
Go To This Time
Warning:
Done
Internet

```
00:00:05:000
13  13  13   0   0   0  13  13  13   0  13  1
1   1   1   0  13  13  13   0  13  13  13  13
```

```
00:00:00:100
00:00:02:500
00:00:05:000
00:00:07:800
```

00:00:08:000

The final state of the nodes in plane 1 after 78 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Done
Internet

Address: http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

Output Files:

```
00:00:07:800
13  13  13   0   0   0  13  13  13   0  13  1
1   1   1   0  13  13  13   0  13  13  13  13
```

Time:

```
00:00:00:100
00:00:02:500
00:00:05:000
00:00:07:800
```

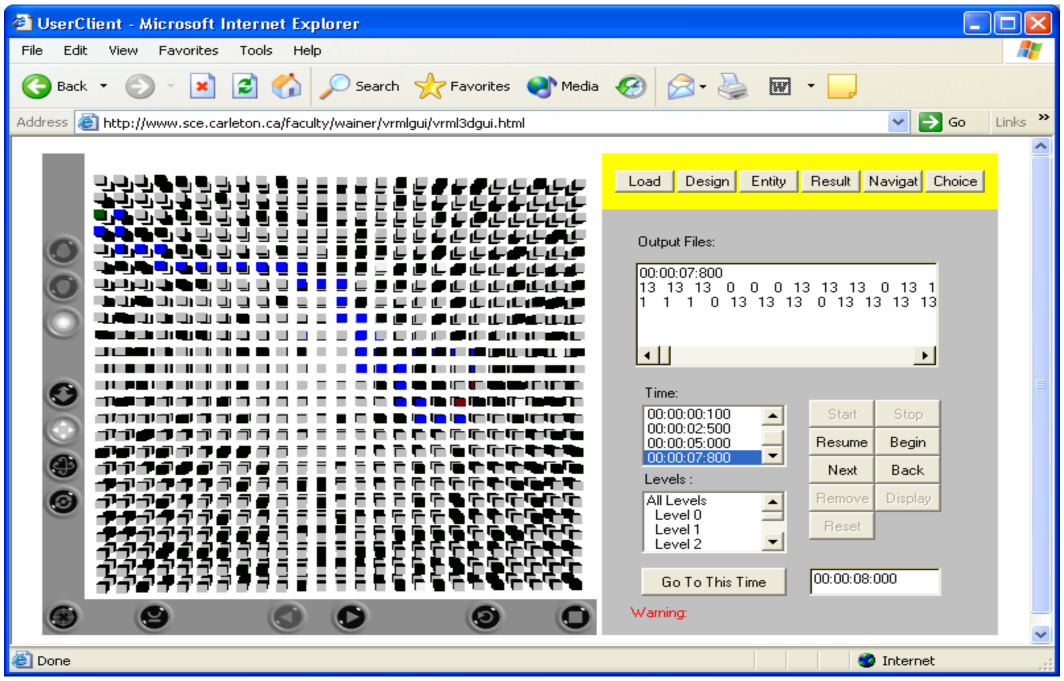
Levels:

```
All Levels
Level 0
Level 1
Level 2
```

00:00:08:000

Warning:

The final state of the nodes in plane 2 after 78 steps of execution is as follows:


UserClient - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Load Design Entity Result Navigat Choice
Start Stop Resume Begin Next Back Remove Display Reset
Go To This Time
Done
Internet

Address: http://www.sce.carleton.ca/faculty/wainer/vrmlgui/vrml3dgui.html

Output Files:

```
00:00:07:800
13  13  13   0   0   0  13  13  13   0  13  1
1   1   1   0  13  13  13   0  13  13  13  13
```

Time:

```
00:00:00:100
00:00:02:500
00:00:05:000
00:00:07:800
```

Levels:

```
All Levels
Level 0
Level 1
Level 2
```

00:00:08:000

Warning:

The results thus show that the model has successfully established the shortest path between the sender and the receiver in each of the planes. The algorithm thus is capable of routing among multiple pairs of senders and receivers simultaneously without having to define more states.

3.3.7 Reaction of the Model to Different Inputs than Those Defined in the Specifications

The requirement of the model is that it has one and only one sender and one and only one receiver node in each plane of the model. The cases that violate this condition have been discussed in detail in Section 3.1.6 and will not be repeated here.

4. CONCLUSIONS

The following conclusions can be drawn from this project:

- If traditional routing algorithms are mapped onto Cell-DEVS, more efficient and more parallel routing algorithms can be found. This project maps the Adaptive On Demand Distance Vector (AODV) protocol for routing in ad hoc networks onto Cell-DEVS. This not only provided insights into the dynamics of the system and its reaction to different input stimuli but also gave an in depth analysis of the protocol for different testing conditions such as the number and location of nodes, connectivity conditions etc. Moreover, this mapping of the algorithm onto Cell-DEVS resulted in the extension of the algorithm in three different directions covered comprehensively in the project. This shows that, generally speaking, mapping of traditional algorithms onto Cell-DEVS can lead into new ideas in theory and implementation of algorithms.

- As a first part of the project, the project extended the original Lee's algorithm to work in 3 dimensions. This extension is straight forward and produces shortest path between two nodes (if one exists) in 3 dimensional space by just using 19 states for each cell. The inherent parallelism in Cell-DEVS is exploited and wave messages and clear state messages are processed concurrently by several nodes. The project only shows how the original algorithm can be extended to the third dimension but the approach taken can be easily extended to any general 'n' number of dimensions.

- The original Lee's algorithm fails for multiple receivers. During the second part the project not only handles multiple receivers but also produces optimal multicast trees. A whole new algorithm has been devised and analyzed. The algorithm is working pretty nicely and many complex scenarios can be solved using this algorithm, producing optimal multicast trees that duplicate the traffic as less as possible and hence save enormous amount of bandwidth. This clearly shows how mapping of routing algorithms onto Cell-DEVS can lead to new ideas and algorithms. The only downside of the algorithm is that it requires that the new nodes be added one by one. They cannot be added simultaneously. However, this problem can easily be handled by having multiple state variables for each cell. This way several nodes can join the multicast tree simultaneously and the parallel computing power of Cell-DEVS would be exploited to its fullest.

- The third part of the project extends the original algorithm to deal with multiple pairs of senders and receivers. The approach taken exploits the parallelism in Cell-DEVS to its fullest and multiple pairs are routed simultaneously. This is done by having multiple planes and assigning one state variable to each of the planes. The approach taken is able to deal with a large number of senders and receivers concurrently.

5. REFERENCES

[1] C. Hochberger and R. Hoffmann, "Solving routing problems with cellular automata". In *Giancarlo Mauri, editor, Cellular Automata in Research and Industry, 1996*, 1996.

[2] Umar Farooq, “Routing in Wireless Ad Hoc Networks”, *Assignment 2 for SYSC 5807*, Systems and Computer Engineering Department, Carleton University, Ottawa, Canada, November 2003.