Modelling spread of worm in LAN using Cell-DEVS

Name: Wubin Ouyang Student I.D: 100934089
Email: Wubin.ouyang@carleton.ca

I. DESCRIPTIONS

This is a model to simulate a typical process of spread of worm, in a LAN. It is implemented in CD++, and can be viewed in CD++ modeler directly. We should admit that there is not a satisfactory definition of a computer virus because this notion has been overloaded with many definitions over the years, and usually it has been mistaken for a Trojan horse or a worm. Nevertheless, one can state that a computer virus is a hidden and malicious program that infects a computer by copying itself to other programs or files. The computer virus is executed when the host program is opened. Then, it searches for uninfected files and tries to attach itself to them too. The action of computer viruses hinders the normal working of computers and includes deleting files, trashing the BIOS, leaving backdoors, spying private data, etc. Consequently, the design of mathematical models that allow one to simulate the computer virus spreading in a computer network is an important issue.

In this simulation, we introduce several states and parameters to make it more realistic. However, parameters may depend on the functionality of a worm and the level of awareness of users. Even the pattern of communication in this LAN may play an important role in the process of spread. Thus, in this simulation, we do not expect to achieve a precise process of spread. Instead, we provide a framework that can adjust to different type of worms and LANs by setting different parameters. And in the following demonstration, several examples will be explained to show this idea.

Several models have been proposed. A popular one is SEIS model, mentioned in *A Computer Virus Spread Model Based on Cellular Automata on Graphs* by *A. Mart´ın del Rey.* In this model, each node/computer of the network can be at one of the following three states: S (susceptible), E (exposed), or I (Infected). Unfortunately, there may be some errors in this paper, and there is no response after my inquiring via E-mail. Another paper written by *A. Mart´ın del Rey* is *A SIR e-Epidemic Model for Computer Worms Based on Cellular Automata.* It presents an S (Susceptible), I (Infected), and R (Recovered) model to simulate the spread of a worm. In recent years, models that involves more states are proposed. A five-state model with S (Susceptible), Q (Questionable), DI (Deadly Infected), I (Infected), and D (Diagnosed) is presented in *Simulation of worm viruses in Network Based on Cellular Automata* (Written in Chinese).

In my modelling, to balance the accuracy and complexity, I will use a four-state model to show the spread of worm, which is composed by S (susceptible), Q (questionable), I (Infected), R (Recovered). Further details are provided below.

II. DEFINITIONS

1. Cell model

The following is the formal definition for the CELL-DEVS model.

CD = < X, Y, I, S, θ, N, d, δint, δext, τ, λ, D >

X = Ø

Y = Ø

S = {-1, 0, 1, 2}

N = { (-1, 0), (0, -1), (0, 1), (1, 0), (0,0), (-1, -1), (-1, 1), (1, -1), (1, 1),(-2, -2), (-2, -1), (-2, 0), (-2, 1), (-2, 2), (-1, -2), (-1, 2), (0, -2), (0, 2), (1, -2), (1, 2), (2, -2), (2, -1), (2, 0), (2, 1), (2, 2) }

d = 100 ms

δint: based on the rules explained later.

2. Two kinds of neighbor

To distinguish influences among different neighbors, we introduce two kinds of neighbor. The first sort is called “adjacent category” which is next to the central cell. Another one called “remote category” which is a little farther. In my model, the “adjacent category” includes cells of { (-1, 0), (0, -1), (0, 1), (1, 0), (0,0), (-1, -1), (-1, 1), (1, -1), (1, 1) }, while the “remote category” includes { (-2, -2), (-2, -1), (-2, 0), (-2, 1), (-2, 2), (-1, -2), (-1, 2), (0, -2), (0, 2), (1, -2), (1, 2), (2, -2), (2, -1), (2, 0), (2, 1), (2, 2) }.

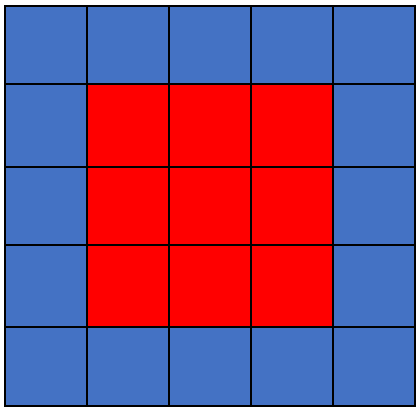


Figure-1 Two kinds of neighbors.

3. Features

In this model, we consider a LAN containing 2500 computers. These computers do not have access to outer network (e.g Internet). The worm is initially rooted in one or several computers, assuming that it is produced deliberately or brought in incautiously. Computers in this LAN have 24 contacts at most, which can be used to spread the worm. 16 of them are in “remote category”, while others are in “adjacent category”. Cells in “adjacent category” have more influences than ones in “remote category”. In this model, this feature is implemented by two macros “Inner_Factor” and “Outer_Factor”.

4. States and transitions.

There are four states in this model.

0: S (Susceptible). S indicates that computer is not infected and immunized. It may be infected in the future, or it may turn into questionable.

-1: R (Recovered). R indicates that computer has been recovered from an infected state or has been immunized and transited from state S. In this state, computers will not be infected again and keep it forever.

1: Q (Questionable). Q indicates that computer is questioned. It comes from situation that some neighbors are infected or questionable. Computer in this state will not spread worm. It is to be recovered and transit its state to R, or to be ruled out to be S.

2: I (Infected). I indicates that computer is infected already. It will spread worm to other computers nearby. In this state, computer keeps its state or gets recovery to R.

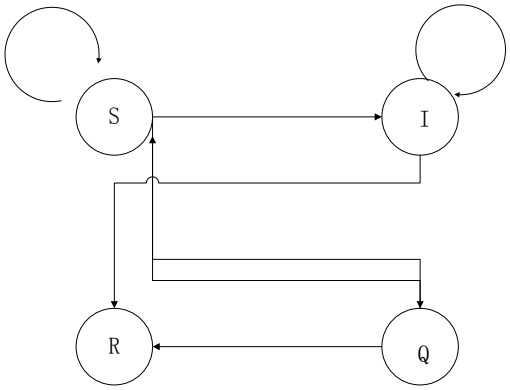


Figure-2 States and transitions.

5. Rules.

Rules are implements on the transitions. These rules consider both the functionality of worm and behavior of users. As mentioned before, we are not expected to commit a precise model. But we can present a realistic framework to implement sorts of worm spread. This model is extendable and easy to modify according to features of other worms.

(1) S->I:

This transition indicates that computer is infected by worm which is spread by other computers. To trigger this event, there must be at least one computer infected among neighbors. The possibility of this transition relates to the numbers of infected computers in “adjacent category” and “remote category”, as well as the “possibility of spreading” that is associated with the mechanism of worm.

The rule in CD++ can be written as below:

```
rule : 2 100  { (0,0)=0 and stateCount(2) > 0 and random <
#macro(Possibility_Of_Spreading) *  ( #macro(Inner_Factor) *
#macro(inner_be2)/8 + #macro(Outer_Factor) * #macro(outer_be2)/16 )  }
```

Where Inner_Factor and Outer_Factor indicate influences of “adjacent category” and “remote category” respectively, and (inner_be2) and (outer_be2) indicate the number of infected computers in “adjacent category” and “remote category”.
This rule also imply that the number of infected computers in neighbor cell play an important role in the process of spread. It complies to the natural process of worm spread.

(2) S->Q
This transition indicates that computer is questionable because of the infecting happened in neighbor cells or increasing number of questionable cells in neighbor. It represents that user notices the danger that computer gets to be infected. User may immunize the computer or rule out it back to S. This transition relates to the number of infected or questionable neighbors, as well as the vigilance of users.
The rule in CD++ is below:

```
rule : 1 100 { (0,0)=0 and ( #macro(Inner_Factor) * #macro(inner_be2) +
#macro(Outer_Factor) * #macro(outer_be2) > 7) and random <
( #macro(Vigilance) * ( #macro(Inner_Factor) * #macro(inner_be2)/8 +
#macro(Outer_Factor) * #macro(outer_be2)/16 ) ) }
rule : 1 100 { (0,0)=0 and ( #macro(Inner_Factor) * #macro(inner_be1) +
#macro(Outer_Factor) * #macro(outer_be1) > 0) and random <
( #macro(Vigilance) * ( #macro(Inner_Factor) * #macro(inner_be1)/8 +
#macro(Outer_Factor) * #macro(outer_be1)/16 ) ) }
```

As can be seen in the rule, only the number of infected or questionable neighbors reaches a threshold, it can be triggered out. This means that an increasing number of infected neighbors can lead user to be more cautious.

(3) Q->R and Q->S
The first transition indicates that computer is immunized by user from the state of questionable. In other words, it will not get infected or spread the worm.
The second transition indicates that computer is ruled out from questionable. User simply ignores the possibility of being infected and does not immunize the computer. These rules are relating to the awareness of users. Here it is implemented by a ratio “Get_Inmune_After_Questioned”.
The rule in CD++ is below:

```
rule : { if ( random < #macro(Get_Inmune_After_Questioned), -1, 0) } 100
{ (0,0)=1 }
```

(4) I->R
This transition indicates that infected computers are rescued and recovered by users. Computers get immune and will not be infected any more. It relates to the responsibility of users to recover their computers and prevent the spread of worm. Considering the increasing number of infected neighbors will drive users to do the recovery, it as well involves the number of infected neighbors.

```
rule : -1 100 { (0,0)=2 and ( #macro(Inner_Factor) * #macro(inner_be2) +
#macro(Outer_Factor) * #macro(outer_be2) > 5) and random <
( #macro(Responsibility) *  ( #macro(Inner_Factor) * #macro(inner_be2)/8 +
#macro(Outer_Factor) * #macro(outer_be2)/16 ) ) }
```

III. SIMULATIONS AND RESULTS

1. Initial settings.

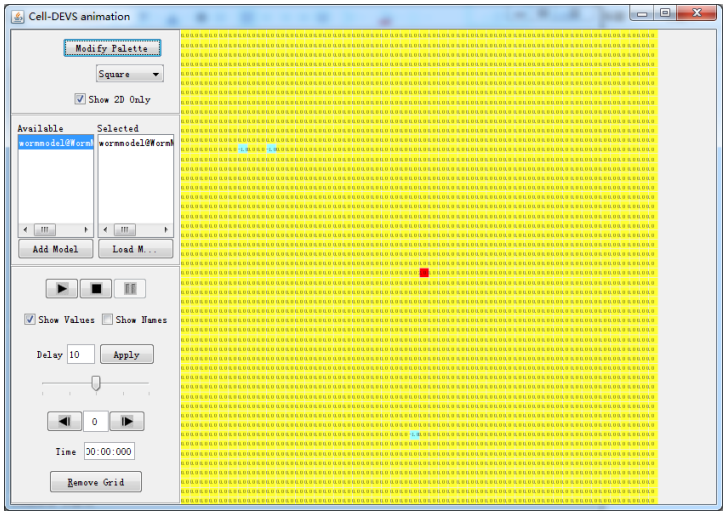
As shown before, in this model or framework, we declare several parameters to adjust different type of worms, networks and even users' behaviors.

To begin this simulation, we need to assign values to those parameters. A bunch of initial settings are shown below:

```
Get_Inmune_After_Questioned): 0.5
Outer_Factor: uniform (0, 0.25)
Inner_Factor: 1
Possibility_Of_Spreading: 0.4
Vigilance: 0.6
Responsibility: 0.7
```

Notice that the "Outer_Factor" which represents the influence made from "remote category" of neighbors is not a constant. This is due to the variance of frequencies and contents in contacts. The "possibility_of_spreading" is related to the functionality of worm, while "Vigilance" and "Responsibility" are relating to the users themselves.

After setting parameters, we now plant worm and some immunized cells among the 2500 computers. As it is shown below:

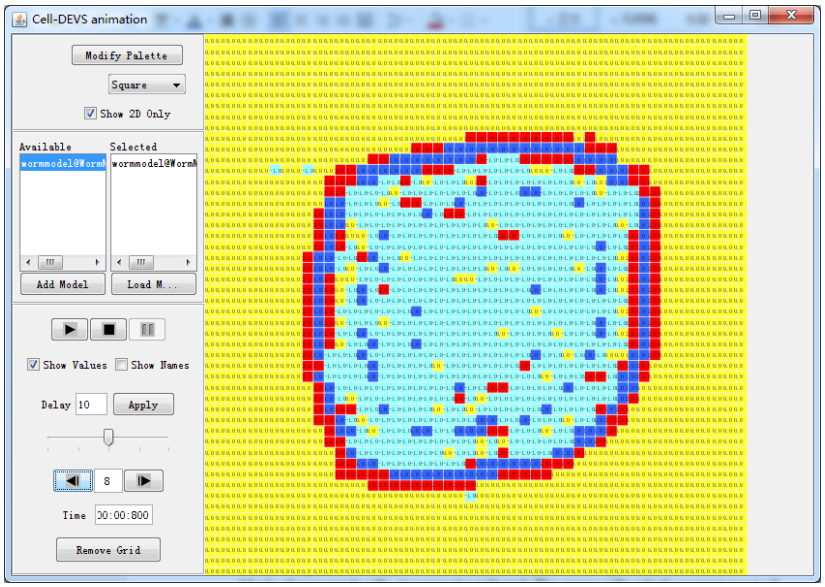


In this graph, red cell is in the state of infected, while yellow one is in state of susceptible and green one is in the state of recovered. Later, a blue cell indicates that the cell is in the state of questionable. Initially, we put one infected cell and three recovered cells which will not get infected in the whole process of simulation.

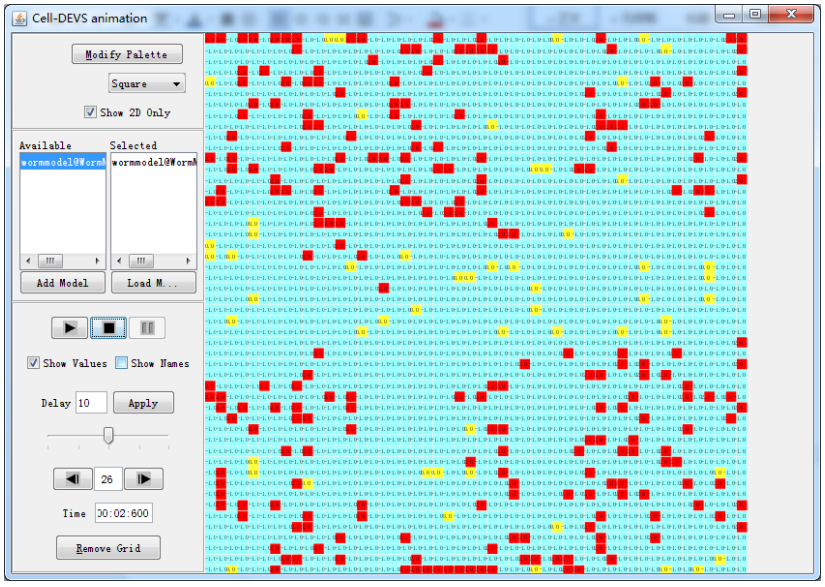
2. Simulation

The simulation under initial settings runs 26 steps to global passive state. The

intermediate and final scenes are shown as below:



Intermediate scene



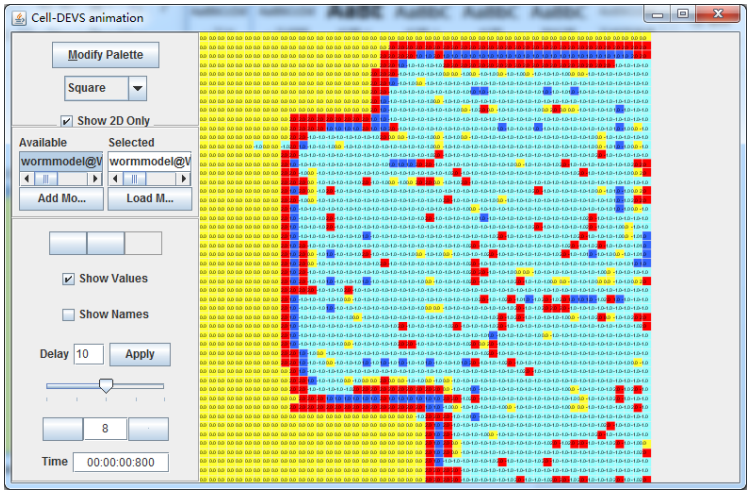
Final scene

In this graph, red cells are still in the state of infected. But because of the large number of recovered cells, it cannot spread the worm any more. The infected cells are then isolated to patches. The yellow cells are in the state of susceptible. Similarly, due to the block of recovered cells, it cannot be infected. The green cells which account most of the 2500 cells are recovered, and this decisively contributes to the end of spread.
A video in attachments has the whole process of simulation.

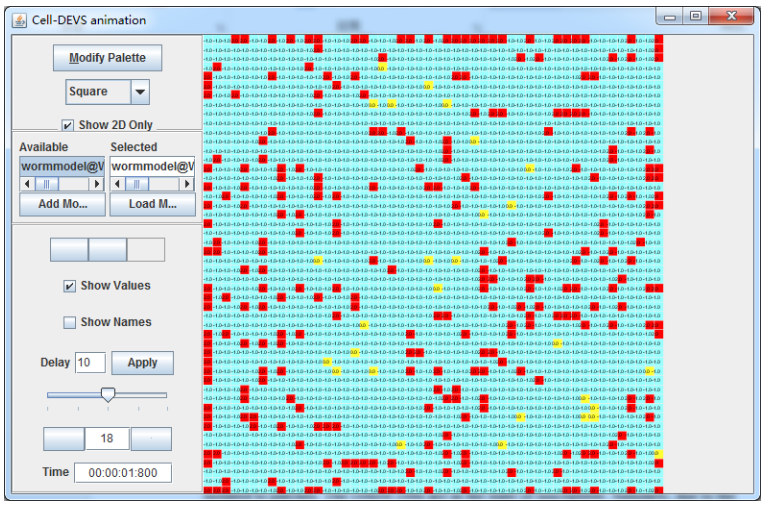
3. Different settings

To test the adjustability of this model, we then modify some parameters.
Assuming that we have a more popular worm that has a larger possibility to spread, we modify the “Possibility_Of_Spreading” to 0.8. Besides, we lower the “Vigilance” and “Responsibility” to indicate that users are not well prepared for this spread.

Finally, we plant three cells with worm to make it spread faster.
The simulation scenes are shown below:



Obviously, worms in this case are spreading faster than they are in the previous settings. It is due to the increased number of worm at very beginning. Besides, it has more infected ones next to recovered cells. This may results from the lower responsibility of users.



The final scene shows that it has more infected cells left than the previous one.

IV. Conclusion

The simulations presented before prove that this model can be used to simulate the process of spread of worm in a LAN. Besides, it shows that it has a strong extendibility and well adjustability. Anyone who wants to reuse this framework only needs to adjust the parameters to specific values.
This model also needs to be updated in some fields. I will do some improvements later and make it more reliable to simulate the process of spread of worm.

Reference:

1. *A Computer Virus Spread Model Based on Cellular Automata on Graphs*. *A. Mart´ın del Rey*
2. *A SIR e-Epidemic Model for Computer Worms Based on Cellular Automata*. *A. Mart´ın del Rey*
3. *Simulation of worm viruses in Network Based on Cellular Automata.*