International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 05 | May 2019 www.irjet.net p-ISSN: 2395-0072

IMPLEMENTATION AND SIMULATION OF FAILSAFE NETWORK ARCHITECTURE

Shikha Saxena[1], Dr. Somnath chandra[2]

[1]PHD student, Mewar University, Chittorgarh, India

Abstract - *In the present scenario distributed computing is become part and parcel of each and every one's life. Computer systems must be arranged in such a fashion so that more and more data transmission can be done with less or no failure in data transmission. A brand new planned fail safe design victimization using star and ring topologies and there implementation in omnet++ simulator is introduced.*

***Key Words*: Topology, omnet++, StRing, star, ring, ned, ini**

1. INTRODUCTION

As the technology is growing distributed computing system comes in to the picture very largely. Growing technology also affect the performance of any distributed system and more interconnected networks also exist there. Network topologies [1-2, 8-9] is the connections of systems physically in different –different manner. Topology can refers to the approach within which the network of computers [5, 6-7] is connected. Every topology is designed as per the need to fulfil some task. LAN is very common example of network topology[3, 4, 8, 9].Local Area Network use to connect various computers and links between them within a specific range. Topology can be defined in two classes Physical topologies [8-9] and Logical topologies [8-9]. Hardware related items like workstations, remote terminals, servers and wiring between assets define by physical topology. Conversely, the illustration of information flow between computers defines by Logical topology. During paper a short description on star, ring and a replacement fail-safe structure are given. The architectural description of physical topologies is defined in section 2. omnet++ simulator and implementation of topologies using omnet++ simulator and how the fail safe architecture works even in case of failure. Conclusions and future work are discussed in section 4.

2. ARCHITECTURAL DESCRIPTION

A distributed embedded system is an embedded system comprised of not one, however many computers. These computers—called computing computers or just computers—cooperate to perform standard functions, doing this by exchanging messages through a network. A system is very reliable if it will perform unceasingly for long periods, while not failing because of faults. A system is adaptation if it will amendment on its own to deal with unpredictable circumstances. The network could need to enable the computers to exchange types of messages, messages that must be exchanged more frequently.[12]

Physical network topology is the physical arrangement of computers in different way like Bus topology, star topology, ring topology.

A. Star Topology: In star topology all the computers are connected by a central hub or switch. Data transmission from one computer to another is happen by that single hub.hob is central controller in star topology .if the hub fails all the systems will stop working.
B. Ring Topology: The ring topology designed in a way that first computer is connected to second ,second connected to third and last computer in network connected to first computer in ring fashion. Data transmission from one computer to another happens through ring. Here the problem is if the failure in any system or link in ring then data transmission will get stop.
C. Proposed fail-safe architecture (StRing topology)-combine both star and ring topology and make a fail-safe structure in case of failure of ring switch will perform the entire task and in case of switch failure ring will be responsible for transmitting the data. , we will design an Ethernet-based communication subsystem that is fault tolerant, that guarantees that messages are exchanged on time, and that allows computers to change the real-time parameters of messages. For instance, the communication subsystem will allow computers to change the frequency with which certain messages are transmitted, what deadlines the messages have to meet, or the volume of data that the messages convey. Moreover—and this will be our main focus—the communication subsystem will ensure that all.

3. BUILDING NETWORK ARCHITECTURE USING OMNET++

To build the different network architecture in omnet++ there is need to define files like NED file, ini file, package declaration and writing sourse program. After creating these file simulation is completed and data transmission started.

a. Create NED file:

In omnet++ first there is a need to construct a topology. Topology is designed using NED file with .ned extension. NED file can be created on text and graphical editor.[10]

With the help of NED language user describes the structure of a simulation model. NED can be defined as Network Description file. Using NED user define simple modules, and these modules can be connect and assemble into compound modules. These compound modules work as network and further use as simulation models.[10]

b. Configure ini file:

For the purpose of execution the Ini file define by user to configure simulation models. It helps in both form-based as well as source editing .The ini File editor supports to configure many documents.

c. Package Declaration

package declaration contain by NED file. Package keyword used by package declaration and specify that package for particular NED file. Default package option is also there. Package contains an unique component type. Package, Directory Structure and the directory of a NED file must match the package declaration.

d. Create C++ program

C++ is used for programming simple modules. Simple module is active module c++ is used to write these modules by the help of simulation class library. Compound module is made by several simple modules. Message definition can be translated into c++ classes.C++ files using extension .cc,.h suffix.[10]

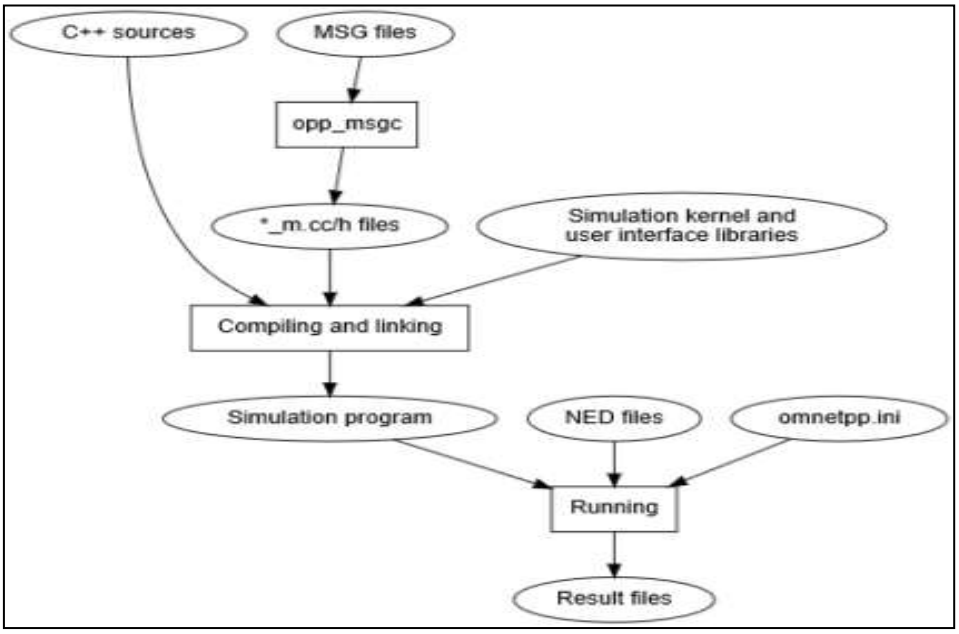

Fig -1: Building and running simulation

3.1. Ring topology simulation in omnet++

To design ring topology first we make ned file. Using ned file graphical approach connect one computer to another computer in such a fashion that last will attach with first computer. After that define package, ini file and write source code .Now built and run the simulation.

Define ned file:

network Net

{

```
submodules:
    computer0: Computer {
        @display("p=200,50");
    }
    computer1: Computer {
        @display("p=306,94");
            }
    computer2: Computer {
        @display("p=350,201");
            }
    computer3: Computer {
        @display("p=306,307");
            }
    computer4: Computer {
        @display("p=199,350");
            }
    computer5: Computer {
        @display("p=93,307");
            }
    computer6: Computer {
        @display("p=50,200");
            }
    computer7: Computer {
        @display("p=93,94");
            }
connections:
    computer0.next <--> DelayChannel <--> computer1.prev;
    computer1.next <--> DelayChannel <--> computer2.prev;
    computer2.next <--> DelayChannel <--> computer3.prev;
    computer3.next <--> DelayChannel <--> computer4.prev;
    computer4.next <--> DelayChannel <--> computer5.prev;
```

```
    computer5.next <--> DelayChannel <--> computer6.prev;
    computer6.next <--> DelayChannel <--> computer7.prev;
    computer7.next <--> DelayChannel <--> computer0.prev;
}
```

b. define package file like omnetpp.org then define ini file

c. Create source code in c++

```
#ifndef COMPUTER_H_
#define COMPUTER_H_
#include <iostream.h>
#include <string.h>
 #include <omnetpp.h>
#include "SimpleMessage_m.h"
class Computer : public cSimpleModule
{
 private:
int counter;
int limit;

protected:
// Algorithm is as follows
 virtual void initialize();
 virtual void handleMessage(cMessage *msg);
 virtual SimpleMessage *generateMessage();
virtual void forwardMessage(SimpleMessage *msg);
};
#endif /* COMPUTER_H_ */
```

d.Write source code (C++ code)

Computer.cc

```
#include "Computer.h"
 // This module class  registered with OMNeT++
```

```
Define_Module(Computer);
 void Computer::initialize()
{
 Variables initialization here.
if (...)
{
ev << "I have index 0, sending initial message" << endl;
 SimpleMessage *msg = generateMessage();
forwardMessage(msg);
 }
}
void Computer::handleMessage(cMessage *msg)
 {
SimpleMessage *smsg = check_and_cast<SimpleMessage *>(msg);
// I am final destination
if (...)
{
ev << "Message " << smsg << " arrived after " << smsg >getHopCount() << " hops.\n";
delete smsg;
// Generate another one.
ev << "Generating another message: ";
SimpleMessage *newmsg = generateMessage();
 forwardMessage(newmsg);
}
else
// I am not final destination
{
forwardMessage(smsg);
 }
}
SimpleMessage* Computer::generateMessage()
```

```
// Produce source and random destination addresses.
// Create message object and set source and destination field.
 SimpleMessage *msg = new SimpleMessage(msgname);
msg->setSource(src);
msg->setDestination(dest);
return msg;
}
void Computer::forwardMessage(SimpleMessage *msg)
{
// Increase hop count
 ...
// Randomly select a next hop k
 ...
ev << "Forwarding message " << msg << " on port out[" << k << "]\n";
send(msg, "out", k);
}
```

When source code is completed the project gets ready to build and run.

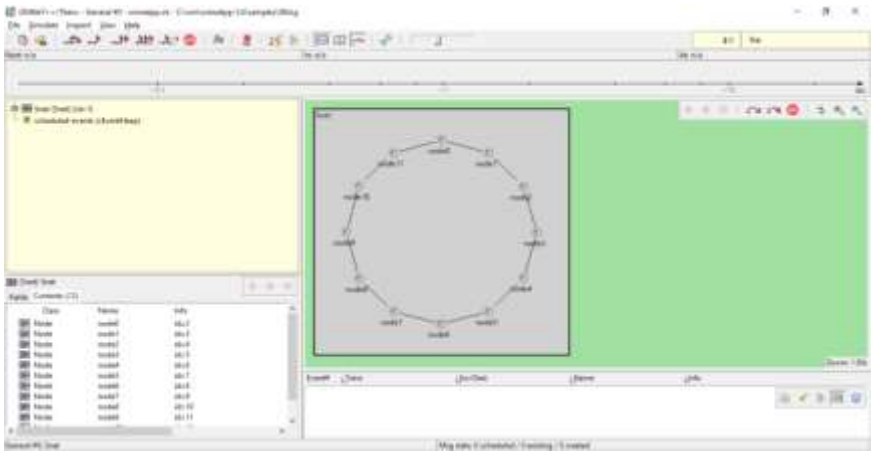

Fig -2: Data Transmission in Ring Topology

Data transmission in ring happens smoothly if there is no fault but if there is any fault occur in link or in computer then data transmission will stop this is weakness of ring topology.

3.2. Star topology simulation

Similarly star topology can be created by defining ned file, package and ini. Here in this structure central device hub is in the middle rest of the devices connected to that hub.

a. NED fie creation

```
import ned.DelayChannel;
module Computer
{
   parameters:
      @display("i=misc/computer_vs;bgb=70,10");
   gates:
      inout switch;
   connections allowunconnected:
}
module Switch
{
   parameters:
      @display("i=misc/computer_vs;bgb=70,10");
   gates:
      inout spoke[];
   connections allowunconnected:
}
//  network with star topology.
network Nstr
{
   submodules:
      switch: Switch {
         @display("p=200,200");
         gates:
            spoke[12]; }
      computer0: Computer {
         @display("p=200,50");  }
      computer1: Computer {
         @display("p=274,71");  }
      computer2: Computer {
```

```
        @display("p=329,126"); }
    computer3: Computer {
        @display("p=350,201");}
    computer4: Computer {
        @display("p=329,275");}
    computer5: Computer {
        @display("p=274,330");}
    computer6: Computer {
        @display("p=199,350");}
    computer7: Computer {
        @display("p=125,330");}
    computer8: Computer {
        @display("p=70,276");}
computer9: Computer {
        @display("p=50,200");}
    computer10: Computer {
        @display("p=70,125");}
    computer11: Computer {
        @display("p=124,71");}
  connections:
    switch.spoke[0] <--> DelayChannel <--> computer0.switch;
    switch.spoke[1] <--> DelayChannel <--> computer1.switch;
    switch.spoke[2] <--> DelayChannel <--> computer2.switch;
    switch.spoke[3] <--> DelayChannel <--> computer3.switch;
    switch.spoke[4] <--> DelayChannel <--> computer4.switch;
    switch.spoke[5] <--> DelayChannel <--> computer5.switch;
    switch.spoke[6] <--> DelayChannel <--> computer6.switch;
    switch.spoke[7] <--> DelayChannel <--> computer7.switch;
    switch.spoke[8] <--> DelayChannel <--> computer8.switch;
    switch.spoke[9] <--> DelayChannel <--> computer9.switch;
    switch.spoke[10] <--> DelayChannel <--> computer10.switch;
```

switch.spoke[11] <--> DelayChannel <--> computer11.switch;

After defining ned file, package and ini files write source code in c++ and build and run, the topology will be shown like figure.

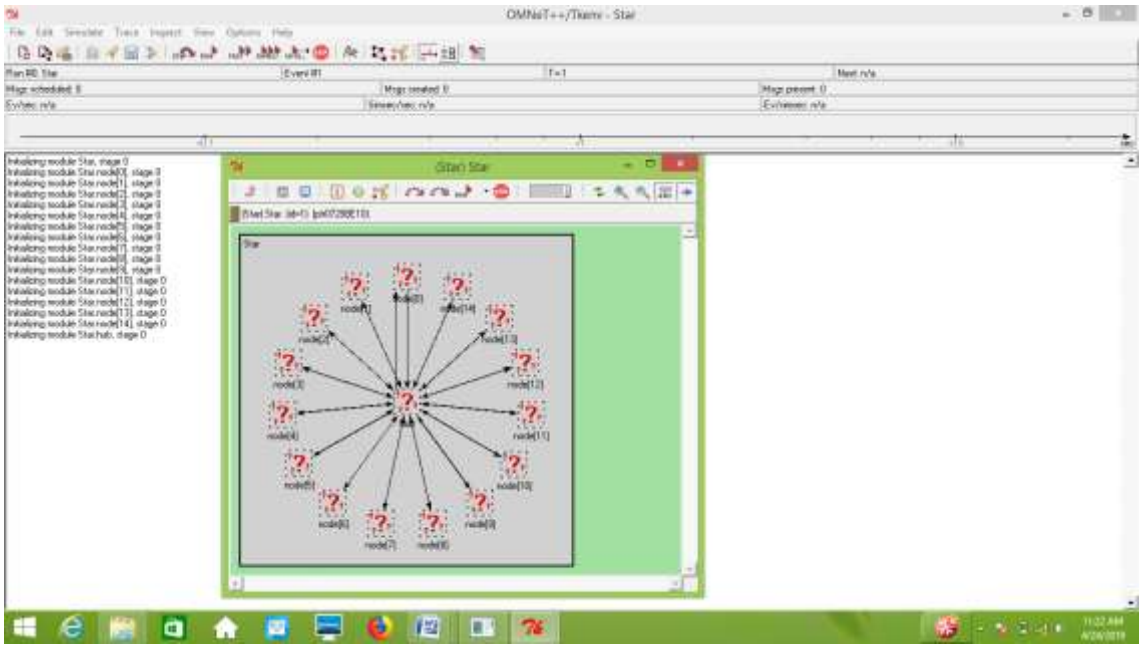

Fig -3: Data Transmission in Star Topology

Star topology data transmission is happen by a central hub if that hub fails all the transmission will get stop .so there must be a structure that can manage such failures and data transmission will not get affected.

3.3. Simulation of fail- safe structure (StRing Topology)

To generate the fail safe structure both ring and star topology arrange in such a manner that if ring fails then switch or hub will perform the routing and is switch fail the ring will perform routing.

a. Create StRing .ned file- as a very first step there is a need to create a ned file where we create a network fail –safe structure.

```
import ned.DelayChannel;

module Computer

{

  parameters:

    @display("i=misc/computer_vs;bgb=68,15;bgl=2");

  gates:

    inout prev;

    inout next;

    inout switch;
```

```
   connections allowunconnected:
}
module switch
{
 parameters:
     @display("i=misc/computer_vs;bgl=2;bgb=68,17");
 gates:
   inout spoke[];
         connections allowunconnected:
}
// network with StRing topology.
network Net1
{
   submodules:
      switch: switch {
         @display("p=200,200");
    gates:
         spoke[12];
      }
      computer0: Computer {
         @display("p=200,50");
     }
      computer1: Computer {
         @display("p=274,71");
      }
      computer2: Computer {
         @display("p=329,126");
      }
      computer3: Computer {
```

```
        @display("p=350,201");
    }
    computer4: Computer {
        @display("p=329,275");
    }
    computer5: Computer {
        @display("p=274,330");
    }
    computer6: Computer {
        @display("p=199,350");
    }
    computer7: Computer {
        @display("p=125,330");
    }
    computer8: Computer {
        @display("p=70,276");
    }
    computer9: Computer {
        @display("p=50,200");
    }
    computer10: Computer {
        @display("p=70,125");
    }
    computer11: Computer {
        @display("p=124,71");
    }
  connections:
switch.spoke[0] <--> DelayChannel <--> computer0.switch;
switch.spoke[1] <--> DelayChannel <--> computer1.switch;
```

switch.spoke[2] <--> DelayChannel <--> computer2.switch;

switch.spoke[3] <--> DelayChannel <--> computer3.switch;

switch.spoke[4] <--> DelayChannel <--> computer4.switch;

switch.spoke[5] <--> DelayChannel <--> computer5.switch;

switch.spoke[6] <--> DelayChannel <--> computer6.switch;

switch.spoke[7] <--> DelayChannel <--> computer7.switch;

switch.spoke[8] <--> DelayChannel <--> computer8.switch;

switch.spoke[9] <--> DelayChannel <--> computer9.switch;

switch.spoke[10] <--> DelayChannel <--> computer10.switch;

switch.spoke[11] <--> DelayChannel <--> computer11.switch;

computer0.next <--> DelayChannel <--> computer1.prev;

computer1.next <--> DelayChannel <--> computer2.prev;

computer2.next <--> DelayChannel <--> computer3.prev;

computer3.next <--> DelayChannel <--> computer4.prev;

computer4.next <--> DelayChannel <--> computer5.prev;

computer5.next <--> DelayChannel <--> computer6.prev;

computer6.next <--> DelayChannel <--> computer7.prev;

computer7.next <--> DelayChannel <--> computer8.prev;

computer8.next <--> DelayChannel <--> computer9.prev;

computer9.next <--> DelayChannel <--> computer10.prev;

computer10.next <--> DelayChannel <--> computer11.prev;

computer11.next <--> DelayChannel <--> computer0.prev;

}

b.Create ini file –create initialization file after .ned file and define package create package file. Now write source code in c++ and build and run, the topology will be shown like figure.

c. Results: a fail –safe structure is ready to work that is the combination of star topology and ring topology.

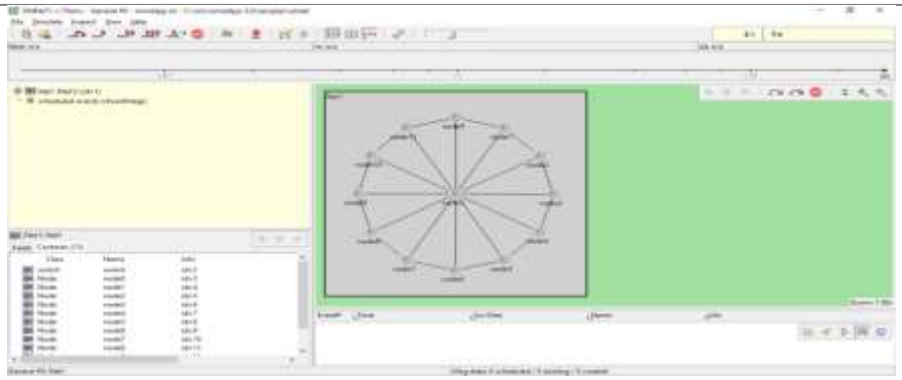

Fig -4: fail safe structure (all node are connected to each

in ring fashion as well as with switch)

In figure(4) the data is transmitted using ring as well as switch if sender computer is adjacent to receiver computer then data will be transmitted using ring otherwise switch. This fail safe structure is work even if ring fails or switch fail. Ring fails then switch will transmit data and if switch fails ring will transmit data.

3.4. Deals with failure

If failure occurs in Switch and it is not able to send data then link of ring is used to transmit data. To represent this transmission the ned file will be changed and further define package and ini and perform some changes in source code. Then build and run simulation.

NED structure:

Fig -5: switch stops working in fail-safe architecture

Result:

Fig -6: Data Transmission through ring in fail-safe architecture

3.5. Switch in working during ring failure

Similarly define Ned structure in such manner that if the link of ring fails the switch performs all the transmission

Result:

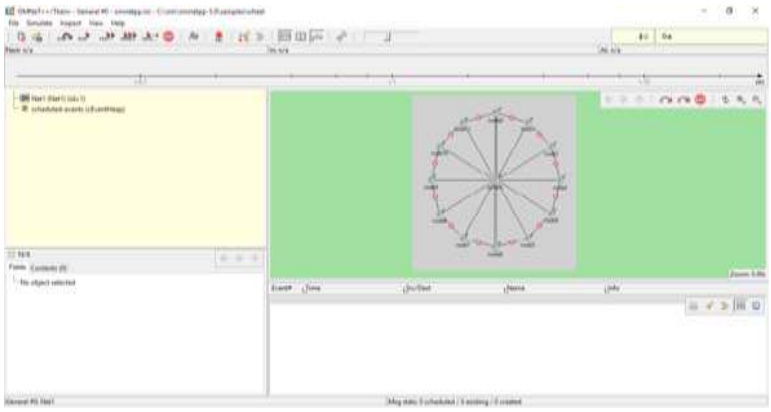

Figure7: Data Transmission through switch if link of ring fail in fail-safe structure

This fail safe structure is very much useful because data is send from source to destination even if there is failure in switch or in ring.

4. CONCLUSION AND FUTURE WORK CONSIDERATION

This paper shows the physical arrangement of systems in a way to generate a fail-safe architecture. This paper also describes a pretty good network simulator tool that is omnet++. Built star, ring and new string topology using OMNeT++ and also show the results when failure occurs. As future work consideration performance assessment can be done for the purpose of testing and assessment of the fail –safe network capabilities, the software model includes several traffic generation patterns besides that scalability of the architecture.

REFERENCES

[1] Banerjee, S., Jain, V., Shah, S., "Regular multihop logical topologies for lightwave networks", Communications Surveys & Tutorials, IEEE, On page(s): 2 - 18 Volume: 2, Issue: 1, First Quarter 1999.

[2] Cem Ersoy, Shivendra PanWar "Topological Design of Interconnected LAN-MAN Networks", IEEE INFOCO, pp. 2260-2269, 1992.

[3] F. Backes, —Transparent Bridges for Interconnection of IEEE 802 LANs,|| IEEE Network, pp. 5-9, January 1988.

[4] Li Chiou Chen "The Impact of Countermeasure Propagation on the Prevalence of Computer Viruses" IEEE Transactions on Systems, MAN, and Cybernetics PartB; Cybernetics Volume 34, Number 2, pp. 823-833, April 2004.

[5] Geon Yoon, Dae Hyun Kwan, Soon Chang Kwon, Yong Oon Park, Young Joon Lee "Ring Topology-based Redundency Ethernet for Industrial Network" SICE-ICASE International Joint Conference, pp. 1404 – 1407, 18-21 Oct. 2006.

[6] Nicholas F. Maxemchuk, Ram Krishnan "A Comparison of Linear and Mesh Technologies---DQDB and Manhattan Street Network" IEEE Journal on Selected Areas in Communications, Volume 11, Number 8, October 1993.

[7] Bannister, J.A., Fratta, L., Gerla, M., "Topological design of the wavelength-division optical network", INFOCOM, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE, On page(s): 1005 - 1013 vol.3, 1990.