Formal Localized Reactive Subnet-based Failure Recovery Model for Sparsely Connected Wireless Sensor and Actor Networks

Hamra Afzaal[1], Nazir Ahmad Zafar[2]
Department of Computer Science
COMSATS Institute of Information Technology
Sahiwal, Pakistan
hamraafzaal@hotmail.edu.pk[1], nazafar@ciitsahiwal.edu.pk[2]

***Abstract*—Formal localized reactive model of Subnet-based Failure Recovery of Sparsely Connected Network for monitoring less critical areas of border is presented in this paper to reduce monitoring by soldiers. It involves Wireless Sensor and Actor Networks (WSANs) partitioned into subnets. The topology of the subnets is sparsely connected in less critical areas where fewer sensors and actors are required. The proposed algorithm is energy efficient as it requires lesser resources in terms of sensors and actors. Further it partitions WSAN into subnets localizing the problem which makes the algorithm energy efficient. Moreover, in the model, the reactive scheme for failure recovery is considered, as in border protection systems the nodes should be replaced when they fail to work. The proposed model is implemented by developing formal specification using VDM-SL which is a formal specification language used for the detailed level analysis of complex systems. That's why the proposed model for border protection is formally implemented using VDM-SL. The developed specification is validated, verified and analyzed using VDM-SL Toolbox.**

***Keywords*—Sparsely Connected Network, WSAN, Border Protection, Failure Recovery, Formal Methods, VDM-SL**

I. Introduction

Every country is concerned with improving the monitoring on the border for protecting the homeland. The traditional border protection systems employ border troops which monitor the specific area. The areas on the border are watched and monitored by border troops according to the pre-planned route in a specific time. The critical areas on the border require constant monitoring whereas less critical areas involve less monitoring. Wireless sensor and actor networks have emerged to operate efficiently and effectively in real-time and mission-critical systems to eliminate or reduce human involvement for providing more accurate results. This paper proposes a model for constant monitoring of the less critical areas of the border by WSANs, because intruders always are in search of areas which are less observed by soldiers.

Though a lot of surveillance schemes [1-3] have been presented which minimize human involvement but still there is a need for a lot of work in this area. This paper presents formal localized reactive model of Subnet-based Failure Recovery of sparsely connected network for border protection in less critical areas. In this model, the WSAN is partitioned into subnets which localize the problem to increase the energy efficiency of the algorithm. The topology of WSAN is sparsely connected which requires fewer number of nodes deployed randomly. In a sparsely connected network, the coverage radius of a node does hold a fewer number of neighbor nodes. The sparsely connected topology is used to save resources because of deployment in less critical areas. The subnets are deployed underground and above ground for monitoring the less critical areas of the border. Different types of sensors detect intrusion and actors track and catch the intruder. The sparsely connected subnets of WSAN are shown using Graph theory model as in Fig. 1. This is because the networks are analogous to graphs. A node represents sensor or an actor and an edge represents wireless communication link among the nodes in the network. Graph theory is used for the semi-formal representation of the informal system to process and store information.

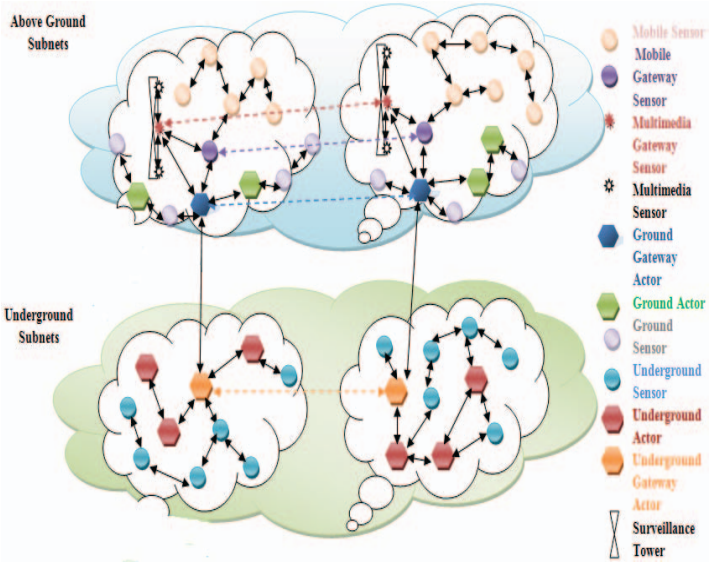

Fig. 1. The sparsely connected subnets of WSAN.

Mostly the existing work focuses on the quantitative performance analysis of non-functional properties through

978-1-4799-7812-0/15/$31.00 ©2015 IEEE

modeling and simulations. However they do not focus on the correctness of the approach. The correctness of the approach is very important in mission-critical and safety-critical systems [4-6]. Formal methods are introduced to transform the informal or semi-formal requirements to a well-defined formal system which confirms about the correctness of the approach. Formal methods are mathematics based techniques used to develop systems in a systematic manner [7]. Formal methods help to analyze critical systems both at abstract and detailed level. VDM-SL is used as a formal specification language to analyze the systems at detailed level. That's why the proposed algorithm for border protection is analyzed by developing formal specification in VDM-SL. The proposed algorithm is validated, verified and analyzed using VDM-SL Toolbox. Rest of the paper is organized as follows. Section II describes problem statement and system model. Some related work is discussed in Section III. The proposed scheme for monitoring less critical areas of the border is presented in Section IV. The formal specification of the proposed scheme is specified in Section V. Results analyzed in Section VI. Finally conclusion of the paper is presented in Section VII.

II. Problem Statement And System Model

Border protection is a critical issue of all countries. Several improvements are made to protect the border but still there is an intensive need to work in this area. The border area is divided into two parts, i.e., critical and less critical. In the critical area of the border constatnt monitoring is required by border troops whereas less critical areas are visited less by the border troops. Conventional border protection systems require extensive human involvement. As humans sometimes cannot provide more accurate reults therefore Wireless Sensor and Actor Networks (WSANs) are used which suit more better for real-time and mission-critical border protection system. WSAN is deployed in the form of sparsely connected subnets which requires fewer number of nodes deployed randomly. In sparsely connected network, the coverage radius of a node does hold fewer number of neighbor nodes. That's why to save resources for the monitoring of less critical area sparsely connected subnets are used. The partitioning of WSAN into subnets localizes the problem which improves the efficiency of the algorithm. Therefore to save resources and for improving the efficiency of the system sparsely connected subnets are deployed. The subnets are deployed under and above ground and different types of nodes are deployed randomly that are far from each other but detect, track and catch the intruder. The failure of some node causes the the intrusion detection process to be stopped. That's why quick recovery is required for the intusion detection process to be continued.

III. Related work

To reduce intensive human involvement in border protection hybrid WSN architecture, *BorderSense*, is presented in [1], which utilizes multimedia and underground sensor networks. This architecture uses multimedia sensors, scalar sensors and mobile sensors. In addition, we have used some gateway nodes and actor nodes in our proposed model to improve this model. The *BorderSense* architecture is for the critical areas of the border. Unlike this our proposed model is for the less critical areas of the border. In the *BorderSense* architecture, nodes are deployed randomly within the sensing range but our model assumes sparsely connected subnets of WSAN which requires fewer nodes deployed randomly and are far from each other. A subnet-based model for backup assigning in WSANs is presented in our previous work [8]. To detect illegal crossing of border unmanned aerial vehicles (UAVs) are presented for aerial surveillance [9], but these require valuable resources of humans for making intelligent decisions for monitoring the border. UAVs are not able to protect the border all the time. Trespassers Favorite Paths (TFPs) notion is introduced in [2] and a tool is provided which is used to forecast the detection probability of a surveillance network with TFPs. Most of the previous work on surveillance techniques is concerned with the analysis through simulations [2]. Simulations do not confirm about the correctness of the approach but are used for the validation of the approach. Formal methods are used for the verification and validation of safety-critical and mission critical systems which confirm about the correctness of the approach. That's why this work uses VDM-SL, a formal specification language for the detailed level analysis of the proposed model for border protection. Some researchers focus on use of formal methods for verification of the algorithms [8, 10-14].

IV. Failure Recovery Model For Border Protection

This section presents algorithm of Localized Reactive Subnet-based Failure Recovery (LRSFR) for sparsely connected WSANs to protect less critical areas of border. It assumes partitioning the WSAN into subnets. The topology in the subnets is sparsely connected. Subnets are deployed underground and above ground. The nodes in the underground subnets are underground sensors, underground gateway actors and underground actors. The nodes in the above ground subnets are ground gateway actors, ground actors and ground sensors, multimedia gateway sensors, multimedia sensors, mobile gateway sensors and mobile sensors. The nodes communicate continuously using their radios and the sparse subnets remain connected in this way. The ground gateway actors and ground actors are mobile robots which catch the intruder and fight with it. The failure of a node may cause the network connectivity to be lost. That's why it is necessary to identify the failed node and to replace it. As in our model, there are different types of nodes therefore the replaced node should match to the type of the failed node. This scheme is reactive as the recovery process is initiated when the failure is detected. Our proposed model is efficient as it requires lesser resources in terms of nodes and network partitioning into subnets localized the problem. In our model, the recovery process is less expensive as it is reactive which is practically applicable. The proactive recovery process becomes expensive and is not practically applicable for border protection critical-system. The pseudo code of the proposed model is presented in Fig. 2.

The failure of a node in the subnet disconnects it from its neighbors. The communication causes lost of the failed node with the neighbor nodes.

LRSFR()
1. WSAN is deployed in the form of Sparsely connected Underground and aboveground subnets on the border areas which requires less monitoring
2. The sparse network does hold less nodes in its coverage radius
3. All type of sensors and actors are assumed as nodes
4. All type of nodes collaboratively detect, track and catch the intruder
5. The following process is used for detecting the intruder
6. Underground sensors detect vibration caused by footsteps of intruder
7. Underground sensors communicate the detected vibration with underground gateway actor which activates underground actors and also communicate with ground gateway actor to confirm detected vibration
8. Ground sensors confirm the detected vibration to ground gateway actor
9. Ground gateway actor transmit the detected information to multimedia gateway sensor
10. Multimedia gateway sensor activates the nearby multimedia sensors for taking image or video of intruder which after detecting the intruder transmit the information back to multimedia gateway sensor
11. Multimedia gateway sensor transmit the intruder information to mobile gateway sensor which activates nearby mobile sensors to track intruder and after tracking intruder transmit information back to mobile gateway sensor
12. Mobile gateway sensor transmit the tracking information to ground gateway actor which activates ground actors to catch the intruder and fight with it
13. If any type of sensor or actor fails the network becomes disconnected and the intrusion detecting process stops
14. That's why failure identification is required for quick recovery
15. The failure is detected by identification of failed nodes
16. The node failure caused the loss of communication among some nodes
17. Once the failed node is identified the recovery process is initiated
18. The recovery process runs separately for each type of node
19. The failed node is replaced only with that particular type of new node which does not belong to the old network
20. The recovery process continues until the network becomes connected

Fig. 2. Pseudo code of Failure Recovery for Border Protection.

For example, the failure of a ground gateway actor is depicted in Fig. 3. The failure of a ground gateway actor causes the loss of its communication with the neighbor ground sensor, ground actor, multimedia gateway sensor and mobile gateway sensor. The failed ground gateway actor in a subnet loses its communication with the other ground gateway actor in the other subnet and with the underground gateway actor in other subnet.

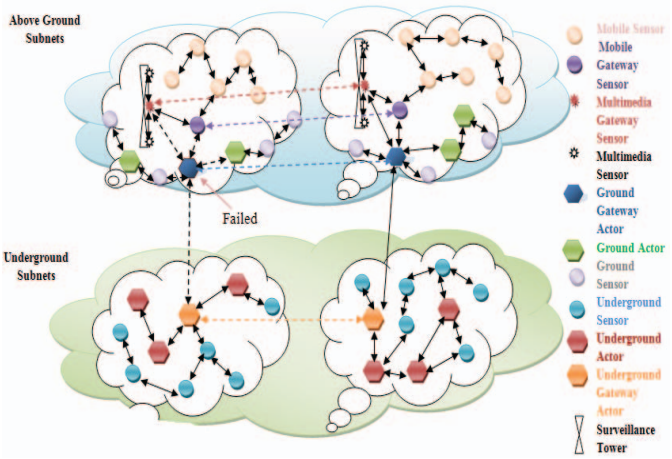

Fig. 3. Example of Failure of Ground Gateway Actor.

V. Formal Specification Of Border Protection

This section presents the formal specification of Localized Reactive Subnet-based Failure Recovery (LRSFR) algorithm for sparsely connected WSANs to protect less critical areas of border. Therefore lesser nodes need to be deployed. For this purpose sparse topology is specified in which nodes are deployed far from each other. The sparse topology is presented as a composite object, *SparseTopology*, which consists of three fields, i.e., *nodes*, *wlinks* and *coverage*. The first field, *nodes*, is the set of Node which is the composite object and have ten fields (*nd_id* for the node identifier, *type* to describe the types of nodes, *status* for checking whether the node is active, passive or failed, *radio* for communication of the nodes, *visible* for checking the visibility, *detect_vibration* for detecting the vibration produced by the footsteps of an intruder, *take* for detecting the intruder by taking an image or video, *communicate* for expressing communication transmission and receiving, *intruderInfo* to be transmitted and received by the nodes, and *mobileRobot* to catch and fight with the intruder). The second field, *wlinks*, of the sparse topology is the set of *Wireless_link* which is also the composite object. Wireless links (edges) represent communication in the network. Two distinct nodes form an edge (wireless link). The third field, *coverage*, of the sparse topology describes that the coverage radius of a node does not hold the next neighbor node.

types
node = ***token***;
Type::*underground_sensor*:*Node*
underground_gateway_actor:*Node*
underground_actor:*Node* *ground_gateway_actor*:*Node*
ground_actor:*Node* *ground_sensor*:*Node*
multimedia_gateway_sensor:*Node* *multimedia_sensor*:*Node*
mobile_gateway_sensor:*Node* *mobile_sensor*:*Node*;

Status=<ACTIVE>|<PASSIVE>|<FAILED>;
Frequency=<SHORT>|<LONG>;
Vibration=<DETECTED>|<NOT_DETECTED>;
Take=<VIDEO>|<IMAGE>;
Transmit=***token***; *Receive*=***token***;
Information:: *t*:*Transmit* *r*:*Receive*;

Node::*nd_idt*:*node* *type*:*Type* *status*: *Status*
radio:*Frequency* *visible*:***bool*** *detect_vibration*:*Vibration*
take:*Take* *communicate*:*Information*
intruderInfo: *Information* *mobileRobot*:*Mobile_Robot*;

Wireless_link::*nod1*:*Node* *nod2*:*Node*
inv mk_*Wireless_link* (*nod1*,*nod2*)==
nod1.nd_idt<>*nod2.nd_idt*;

Coverage::*minRange*:***nat*** *maxRange*:***nat***;

SparseTopology::*nodes*:***set of*** *Node*
wlinks:***set of*** *Wireless_link* *coverage*:***set of*** *Coverage*
inv mk_*SparseTopology*(*nodes*, *wlinks*,*coverage*) ==
forall *wl* ***in set*** *wlinks* & *wl.nod1* ***in set*** *nodes* ***and***

```
  wl.nod2 in set nodes and forall nod in set nodes &
  (exists wl in set wlinks & (nod=wl.nod1 and nod=wl.nod2))
  and (forall n1, n2 in set coverage &
  (n1.maxRange<=n2.minRange));
```

Invariants: (1) All wireless links employs two nodes. (2) There is a wireless link for all nodes in the sparse topology. The isolated node is not the component of the sparse network.

Two types of subnets of WSAN are used in this model, i.e., underground and above ground subnets. The underground subnet is specified as a composite object, *UnderGroundSubnet*. It is defined by four fields. The first field *ugs* explains that the underground subnet is basically a sparse network. The second field *nodes* is required as the underground subnet employs nodes. The *wlinks* is used as communication between nodes is wireless. The fourth field *visible* shows the visibility status.

```
UnderGroundSubnet::ugs:set of SparseTopology
  nodes:set of Node  wlinks:set of Wireless_link  visible:bool
inv mk_UnderGroundSubnet(ugs,nodes,-, visible)==
  forall subnet in set ugs & (card subnet.nodes>=3) and
  visible=false and forall nd in set nodes &
  (nd.type.underground_sensor in set nodes and
  nd.type.underground_gateway_actor in set nodes and
  nd.type.underground_actor in set nodes);
```

Invariants: (1) Three types of nodes exist in the underground subnet. (2) The nodes are not visible to intruder. (3) Underground sensor, gateway actor and actor nodes exist in the underground subnet.

The above ground subnet is presented as a composite object, *AboveGroundSubnet*. It is defined by five fields. The first field *ags* illustrates that the above ground subnet is basically a sparse network. The second field *nodes* is required as the above ground subnet employs nodes. The third field *wlinks* is used as communication between nodes is wireless. The fourth field *sts* is the set of *Surveillance_Tower* which is also a composite object. Multimedia gateway sensor and multimedia sensors are deployed on surveillance tower. The neighbor surveillance tower in the other subnet is within communication range of a surveillance tower in a subnet. The fifth field *visible* shows the visibility status.

```
Signals=<Transmit>|<Receive>;
Surveillance_Tower:: multimedia_gateway_sensor:Node
  multimedia_sensor:Node    coverage:set of Coverage
  sgl:Signals  inv mk_Surveillance_Tower(-,-,coverage,-) ==
  forall t1,t2 in set coverage & (t1.maxRange>=t2.minRange);

AboveGroundSubnet::ags: set of SparseTopology
    nodes:set of Node      wlinks:set of Wireless_link
    sts: set of Surveillance_Tower  visible:bool
inv mk_AboveGroundSubnet(ags, nodes,-,-,visible)==
   forall subnet in set ags & (card subnet.nodes>=7) and
   visible=true and forall nd in set nodes &
   (nd.type.ground_gateway_actor in set nodes and
    nd.type.ground_actor in set nodes and
   nd.type.ground_sensor in set nodes and
   nd.type.multimedia_gateway_sensor in set nodes and
   nd.type.multimedia_sensor in set nodes and
   nd.type.mobile_gateway_sensor in set nodes and
   nd.type.mobile_sensor in set nodes) ;
```

Invariants: (1) The above ground subnet nodes are visible. (2) Gateway actor, ground actor and sensor, multimedia gateway sensor, multimedia sensor, mobile gateway sensor and mobile sensor are deployed above the ground.

The intruder is defined by a composite object, *Intruder*. It has three fields. The first field *i_identifier* describes that every intruder is identified by its unique identifier. The second field *des* exhibits that intruder can destroy multimedia sensors and ground sensors. The third field information is used for the transmission and receiving of information.

```
String=seq of char;
Destroy::multimedia_sensor:set of Node
          ground_sensor:set of Node;
Intruder::i_identifier: String          des: Destroy
   information:Information;
```

The mobile robot is used for detecting the intruder. It is presented as a composite object, *Mobile_Robot*. It has four fields. The first field *m_identifier* describes that every mobile robot has a unique identifier. The second field *opr* expresses the operation of the mobile robot that is idle, failed or working. The third field *region* describes that robot work in the affected region. The fourth field *take_action* is used to describe the action of the mobile robot that is to catch and fight intruder.

```
Operate=<IDLE>|<FAILED>|<WORK>;
Region=<AFFECTED>|<NOT_AFFECTED>;
Take_Action=<CATCH_INTRUDER>|<FIGHT>;
Mobile_Robot::m_identifier:Node    opr:Operate
                region:Region      take_action:Take_Action;
```

The state space of border is specified by *BorderProtection.* It consists of eighteen attributes. The first, second and third attributes, i.e., nodes, wlinks and coverage are used for defining the sparse topology. The fourth and fifth attributes, i.e., ugs and ags represent underground and above ground subnets. The sixth, seventh and eighth attributes, i.e., ugsensor, ugatewayactor and ugactor present underground sensor, underground gateway actor and underground actor respectively. The remaining attributes, *ggatewayactor*, *gactor*, *gsensor*, *mulgatewaysensor*, *mulsensor*, *mobgatewaysensor*, *mobsensor*, *mobile_robots*, *intruders* and *path* describe ground gateway actor, ground actor, ground sensor, multimedia gateway sensor, multimedia sensor, mobile gateway sensor, mobile sensor, mobile robots, intruders and communication path respectively.

```
state BorderProtection of        nodes:set of Node
   wlinks:set of Wireless_link   coverage:set of Coverage
   ugs:set of UnderGroundSubnet
   ags: set of AboveGroundSubnet   ugsensor: set of Node
  uggatewayactor:set of Node        ugactor:set of Node
  ggatewayactor: set of Node       gactor: set of Node
```

```
  gsensor: set of Node            mulgatewaysensor: set of Node
  mulsensor: set of Node          mobgatewaysensor: set of Node
  mobsensor: set of Node          mobile_robots:set of Node
  intruders:set of Intruder       path:seq of node
inv mk_BorderProtection(nodes,wlinks,-,ugs,ags,-,-,-,
ggatewayactor, gactor,-,-,-,-,-,mobile_robots,-, path) ==
    forall usubnet in set ugs &(card usubnet.nodes>=3 and
    forall nd in set nodes & (nd.type.underground_sensor
    in set (nodes and nd.type.underground_gateway_actor
    in set nodes and nd.type.underground_actor in set nodes
    and (exists wl in set wlinks &
      (nd.type.underground_sensor = wl.nod1 and
       nd.type.underground_sensor = wl.nod2 and
       nd.type.underground_sensor = wl.nod1 and
       nd.type.underground_gateway_actor = wl.nod2 and
       nd.type.underground_gateway_actor = wl.nod1 and
       nd.type.underground_actor = wl.nod2 and
       nd.type.underground_gateway_actor = wl.nod1 and
       nd.type.underground_gateway_actor = wl.nod2 and
       nd.type.underground_actor= wl.nod1 and
       nd.type.underground_actor = wl.nod2)))))
       and forall asubnet in set ags & (card asubnet.nodes>=7
       and (forall nd in set nodes &
       (nd.type.ground_gateway_actor in set nodes and
        nd.type.ground_actor in set nodes and
        nd.type.ground_sensor in set nodes and
        nd.type.multimedia_gateway_sensor in set nodes and
        nd.type.multimedia_sensor in set nodes and
        nd.type.mobile_gateway_sensor in set nodes and
        nd.type.mobile_sensor in set nodes and
       (exists wl in set wlinks &(nd.type.ground_gateway_actor
         = wl.nod1 and nd.type.ground_actor = wl.nod2 and
        nd.type.ground_actor = wl.nod1 and
        nd.type.ground_actor = wl.nod2 and
        nd.type.ground_actor = wl.nod1 and
       nd.type.ground_sensor = wl.nod2 and
       nd.type.ground_sensor = wl.nod1 and
       nd.type.ground_sensor = wl.nod2 and
       nd.type.ground_gateway_actor = wl.nod1 and
       nd.type.ground_sensor = wl.nod2 and
       nd.type.ground_gateway_actor = wl.nod1 and
       nd.type.ground_gateway_actor = wl.nod2 and
       nd.type.ground_gateway_actor = wl.nod1 and
       nd.type.multimedia_gateway_sensor = wl.nod2 and
       nd.type.multimedia_gateway_sensor = wl.nod1 and
       nd.type.multimedia_sensor = wl.nod2 and
       nd.type.multimedia_sensor = wl.nod1 and
       nd.type.multimedia_sensor = wl.nod2 and
       nd.type.mobile_gateway_sensor = wl.nod1 and
       nd.type.mobile_sensor = wl.nod2 and
       nd.type.mobile_sensor = wl.nod1 and
       nd.type.mobile_sensor = wl.nod2 and
       nd.type.mobile_gateway_sensor = wl.nod1 and
       nd.type.mobile_gateway_sensor = wl.nod2 and
       nd.type.mobile_gateway_sensor = wl.nod1 and
       nd.type.ground_gateway_actor = wl.nod2)))))and
    forall m in set mobile_robots & (exists g in set gactor &
       (exists gg in set ggatewayactor &(m.nd_idt=g.nd_idt and
        m.nd_idt= gg.nd_idt)))and forall i in set inds path &
        (exists wl in set wlinks & (path(i)= wl.nod1.nd_idt and
          path(i+1)=  wl.nod2.nd_idt))
init b==b=mk_BorderProtection({},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},[])
end
```

Invariants: (1) For all the underground subnets three types of nodes exist in the underground subnet, i.e., underground sensor, underground gateway actor and underground actor nodes. (2) There exists wireless links between all underground sensors. (3) There exists wireless links between underground sensors and underground gateway actor. (4) A wireless link exists between underground gateway actor and underground actor. (5) All the underground gateway actors are connected through wireless links. (6) There exists wireless links between all underground actors. (7) The above ground subnet employs seven nodes, i.e., ground gateway actor, ground actor, ground sensor, multimedia gateway sensor, multimedia sensor, mobile gateway sensor and mobile sensor. (8) There exists wireless link between ground gateway actor and ground actor. (8) There also exists wireless links between all ground actors. (9) Ground actor and ground sensor communicate through wireless link. (10) There must be wireless links between all ground sensors. (11) Wireless links exist between ground sensor and ground gateway actor. (12) All ground gateway actors are connected with each other through wireless link. (13) There also exists wireless link between ground gateway actor and ground multimedia sensor. (14) Wireless link also exists between multimedia gateway sensors. (15) All multimedia sensors also have wireless links between them. (16) Mobile gateway sensors communicate with mobile sensors through wireless links. (17) All the mobile sensors communicate with each other. (18) The mobile gateway sensors communicate through wireless link. (19) The mobile gateway sensors are connected with ground gateway actors through wireless links. (20) Mobile robots are ground actor and ground gateway actor. (21) The nodes are connected as communication path exists between it.

Initialization: All the attributes are initialized respectively.

Now the formal specification of the operations is described below. Firstly formal specification of *ProtectBorder* is presented. The attributes *nodes*, *wlinks*, *coverage*, *ugs* and *ags* are being read in this operation.

```
operations
ProtectBorder()
ext rd nodes:set of Node
     rd wlinks:set of Wireless_link
     rd coverage:set of Coverage
     rd ugs:set of UnderGroundSubnet
     rd ags: set of AboveGroundSubnet
pre true
post forall u in set ugs & (exists s in set u.nodes &(
 s.type.underground_sensor.detect_vibration=<DETECTED>
 and s.type.underground_gateway_actor.communicate.r =
```

```
s.type.underground_sensor.communicate.t and
s.type.ground_gateway_actor.communicate.r=
s.type.underground_gateway_actor. communicate.t and
s.type.ground_gateway_actor. communicate.r=
s.type.ground_sensor.communicate.t and
s.type.multimedia_gateway_sensor.communicate.r=
s.type.ground_gateway_actor.communicate.t and
s.type.multimedia_gateway_sensor.communicate.r=
s.type.multimedia_sensor.communicate.t and
s.type.multimedia_sensor.take=<IMAGE> or
s.type.multimedia_sensor.take=<<VIDEO>> and
s.type.multimedia_gateway_sensor.communicate.r=
 s.type.multimedia_sensor.communicate.t and
 s.type.mobile_gateway_sensor.communicate.r=
 s.type.multimedia_gateway_sensor.communicate.t and
 s.type.mobile_sensor.communicate.r =
 s.type.mobile_gateway_sensor.communicate.t and
 s.type.mobile_gateway_sensor.intruderInfo.r=
 s.type.mobile_sensor.intruderInfo.t and
 s.type.ground_gateway_actor.intruderInfo.r=
 s.type.mobile_gateway_sensor.intruderInfo.t and
 s.mobileRobot.take_action=<CATCH_INTRUDER> and
 s.mobileRobot.take_action=<FIGHT>));
```

Pre/Post Conditions: (1) The underground sensors detect vibration produced by intruder footsteps and communicate the information with underground gateway actor. (2) The underground gateway actors communicate the information with ground gateway actor. (3) The ground sensors confirm the vibration information to ground gateway actor. (4) The ground gateway actors transmit the information to multimedia gateway sensor. (5) The multimedia gateway sensor communicate with the nearby multimedia sensors to take the image or video of the intruder and the multimedia sensors after detecting transmit the information back to multimedia gateway sensor. (6) The multimedia gateway sensors communicate with mobile gateway sensor to track the intruder. (7) The mobile gateway sensors communicate with the nearby mobile sensors to track the intruder. (8) The mobile sensors communicate the intruder tracked information with mobile gateway sensor and the mobile gateway sensors transmit it to ground gateway actor.

Now Formal specification of *FailureIdentification* the operation is described. This operation is specified in general for all types of nodes. It takes the failed node as input. The nodes and wlinks are being written and path is being read.

```
FailureIdentification(failed:Node)
ext wr nodes:set of Node      wr wlinks:set of Wireless_link
    rd path: seq of node
pre failed in set nodes
post nodes=nodes~ \ {failed}and
     wlinks= wlinks ~ \{wl|wl in set wlinks &
     (wl.nod1=failed and wl.nod2 in set nodes)}and
     forall i in set inds path & (exists wl in set wlinks &
     (path(i)<> wl.nod1.nd_idt and
      path(i+1)<> wl.nod1.nd_idt)) ;
```

Pre/Post Conditions: (1) Initially the failed node should be the part of nodes of the network. (2) The sparse topology is updated by removing the failed node from the network. (3) The wireless links are updated by removing those that incident at the failed node. There should not exist a path between some nodes which confirm the failure.

Once the failure is detected the recovery process is initiated. We have modeled the recovery process for all ten types of nodes. But due to lack of space we present only the recovery process for four types of nodes.

The *UndergroundGatewayActorRecovery* operation is presented. It takes input the failed and replaces a new one. The attributes, *nodes*, *ugsensor* (underground sensor) and *ugactor* (underground actor) are being read and *wlinks* and *ugatewayactor* (underground gateway actor) are being written.

```
UnderGroundGatewayActorReocvery (failed:Node,
replaced:Node)
ext rd nodes:set of Node      wr wlinks:set of Wireless_link
    rd ugsensor: set of Node  wr uggatewayactor:set of Node
    rd ugactor:set of Node
pre failed in set uggatewayactor and
     replaced not in set uggatewayactor
post replaced in set uggatewayactor ~\{failed}and
      uggatewayactor = uggatewayactor ~ union {replaced}
      and wlinks= wlinks ~ union{wl|wl in set wlinks &
      (wl.nod1=replaced and wl.nod2 in set nodes)}
      and forall wl in set wlinks &(exists ugga1 in set
      uggatewayactor ~ & (exists ugs1 in set ugsensor &
      (exists uga1 in  set ugactor & (wl.nod1=replaced and
      (wl.nod2.nd_idt=ugga1.nd_idt  or
       wl.nod2.nd_idt=ugs1.nd_idt  or
       wl.nod2.nd_idt=uga1.nd_idt))));
```

Pre/Post Conditions: (1) Initially the failed node should belong to the set of underground gateway actor and the node to be replaced should not be the part of the network or should not belong to the old underground gateway actors. (2) The new node should be put in place of the failed node. (3) The new node now becomes the part of the underground gateway actors. The wireless links are updated by the addition of new node in the edge. (4) The replaced node (underground gateway actor) is on one side of the wireless link and on the other side there may be old underground gateway actor, or underground sensor, or underground actor.

Now the operation, *GroundGatewayActorRecovery*, is illustrated. It takes input the failed and replaces a new node. The attributes, *nodes*, *gactor* (ground actor) and *gsensor* (ground sensor) are being read and *wlinks* and *ggatewayactor* (ground gateway actor) are being written.

```
GroundGatewayActorRecovery(failed:Node, replaced:Node)
ext rd nodes:set of Node    wr wlinks:set of Wireless_link
    wr ggatewayactor: set of Node   rd gactor: set of Node
    rd gsensor: set of Node
pre failed in set ggatewayactor and
     replaced not in set ggatewayactor
```

```
post replaced in set ggatewayactor ~\{failed}and
      ggatewayactor = ggatewayactor ~ union {replaced}
     and wlinks= wlinks ~ union{wl|wl in set wlinks &
     (wl.nod1=replaced and wl.nod2 in set nodes)}
     and forall wl in set wlinks &(exists gga1 in set
     ggatewayactor~ & (exists gs1 in set gsensor &
    (exists ga1 in set gactor & (wl.nod1=replaced and
    (wl.nod2.nd_idt=gga1.nd_idt or
     wl.nod2.nd_idt=gs1.nd_idt or
     wl.nod2.nd_idt=ga1.nd_idt)))));
```

Pre/Post Conditions: (1) Initially the failed node must be in the set of ground gateway actor and the node to be replaced must not be the part of the network or it should not belong to the old ground gateway actors. (2) The new node must be put in place of the failed node. (3) The new node now becomes the part of the ground gateway actors. The wireless links are updated by the addition of a new node in the edge. (4) The replaced node (underground gateway actor) is on one side of the wireless link and on the other side there may be old ground gateway actor, or ground sensor, or ground actor.

Now the operation, *MultimediaGatewayActorRecovery*, is depicted. It takes input the failed and replaces a new node. The attributes, *nodes*, *mulsensor* (multimedia sensor) and *mobgatewaysensor* (mobile gateway sensor) are being read and *wlinks* (wireless links) and *mulgatewaysensor* (multimedia gateway sensor) are being written.

```
MultimediaGatewaySensorRecovery(failed:Node,
replaced:Node)
ext rd nodes:set of Node   wr wlinks:set of Wireless_link
    wr mulgatewaysensor: set of Node
    rd mulsensor: set of Node
    rd mobgatewaysensor: set of Node
pre failed in set mulgatewaysensor and
     replaced not in set mulgatewaysensor
post replaced in set mulgatewaysensor~\{failed}and
    mulgatewaysensor = mulgatewaysensor~ union {replaced}
   and wlinks= wlinks ~ union{wl|wl in set wlinks &
   (wl.nod1=replaced and wl.nod2 in set nodes)} and
   forall wl in set wlinks &(exists mgs1 in set
   mulgatewaysensor~ & (exists ms1 in set mulsensor &
  (exists mogs1 in set mobgatewaysensor &
  (wl.nod1=replaced and  (wl.nod2.nd_idt=mgs1.nd_idt  or
   wl.nod2.nd_idt=ms1.nd_idt  or
   wl.nod2.nd_idt=mogs1.nd_idt)))));
```

Pre/Post Conditions: (1) Initially the failed node should belong to the set of multimedia gateway sensor and the node to be replaced should not the part of the network or it should not belong to the old multimedia gateway sensors. (2) The new node must be put in place of the failed node. (3) The new node now becomes the part of the multimedia gateway sensors. The wireless links are updated by the addition of a new node in the edge (wireless link). (4) The replaced node (mobile gateway sensor) is on one side of the wireless link (edge) and on the other side there may be old multimedia gateway sensor, or multimedia sensor, or mobile gateway sensor.

The operation, *MobileGatewayActorRecovery*, is specified. It takes input the failed and replaces a node. The attributes, *nodes*, *mulgatewaysensor* (multimedia gateway sensor), *mobsensor* (mobile sensor) and *mobgatewaysensor* (mobile gateway sensor) are being read and wireless links and mobile gateway sensor are being written.

```
MobileGatewaySensorRecovery(failed:Node, replaced:Node)
ext rd nodes:set of Node    wr wlinks:set of Wireless_link
 rd mulgatewaysensor: set of Node rd mobsensor:set of Node
 wr mobgatewaysensor:set of Node
 rd ggatewayactor:set of Node
pre failed in set mobgatewaysensor and
     replaced not in set mobgatewaysensor
post replaced in set mobgatewaysensor~\{failed}and
   mobgatewaysensor = mobgatewaysensor~ union {replaced}
   and wlinks= wlinks ~ union{wl|wl in set wlinks &
   (wl.nod1=replaced and wl.nod2 in set nodes)}and
   forall wl in set wlinks &(exists mogs1 in set
   mobgatewaysensor~ & (exists mgs1 in set
   mulgatewaysensor & (exists mos1 in set mobsensor &
   (exists gga1 in set ggatewayactor &(wl.nod1=replaced and
   (wl.nod2.nd_idt=mogs1.nd_idt  or
    wl.nod2.nd_idt=mgs1.nd_idt or
    wl.nod2.nd_idt=mos1.nd_idt or
    wl.nod2.nd_idt=gga1.nd_idt))))));
```

Pre/Post Conditions: (1) Initially the failed node should belong to the set of mobile gateway sensor and the node to be replaced should not be the part of the network or it should not belong to the old mobile gateway sensors. (2) The new node must be put in place of the failed node. (3) The new node now becomes the part of the mobile gateway sensors. The wireless links are updated by the addition of a new node in the edge. (4) The replaced node (mobile gateway sensor) is on one side of the wireless link and on the other side there may be old mobile gateway sensor, or multimedia gateway sensor, or mobile gateway sensor or ground gateway actor.

VI. RESULTS AND ANALYSIS

In this section model analysis of the proposed algorithm for border protection is done. Analyzing the specification through computer tools ensures its correctness. That's why the specification of Localized Reactive Subnet-based Failure Recovery (LRSFR) algorithm is analyzed through VDM-SL Toolbox. A model analysis snapshot of the proposed model is presented in Fig. 4. It is analyzed that the proposed model does not have any syntax or type error. Although some errors cannot be detected by syntax or type checker that's why dynamic checking is enabled for checking actual types during run time. The integrity examiner is used for generating integrity properties. The integrity properties present conditions under which no run time error should occur. If the integrity properties evaluates to true then there is no run time error in the specification. It is observed in this specification that all integrity

properties evaluates to true. The syntax and type checker evaluates the whole specification and integrity examiner evaluates the state and operations as presented in TABLE I.

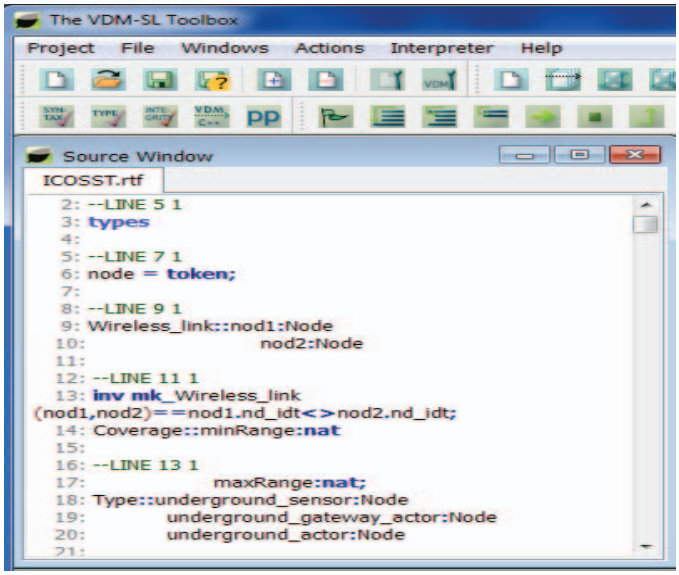

Fig. 4. Model Analysis of the proposed model.

TABLE I. ANALYSIS OF STRUCTURES, STATE AND OPERATIONS.

Structures, State and Operations	Syntax Check	Type Check	Integrity Check
Wireless_link	Y	Y	-
SparseTopology	Y	Y	-
UnderGroundSubnet	Y	Y	-
AboveGroundSubnet	Y	Y	-
Intruder	Y	Y	-
Mobile_Robot	Y	Y	-
BorderProtection	Y	Y	Y
ProtectBorder	Y	Y	Y
FailureIdentification	Y	Y	Y
UnderGroundGatewayActorRecovery	Y	Y	Y
GroundGatewayActorRecovery	Y	Y	Y
MultimediaGatewaySensorRecovery	Y	Y	Y
MobileGatewaySensorRecovery	Y	Y	Y

VII. CONCLUSION

This paper presented a Formal Localized Reactive model of Subnet-based Failure Recovery of sparsely connected network for monitoring of less critical areas of border. The topology of WSANs is sparsely connected as fewer sensors and actors are required in less critical areas. But still there is a need for monitoring in less critical areas because intruder can find areas which are less monitored by soldiers. That's why to reduce monitoring by soldiers sparsely connected WSANs are deployed. This model also considers the reactive scheme for the recovery of failed node as failure of a node may cause the intrusion detection process to be stopped. The sparsely connected subnets of WSAN are presented graphically for the semiformal representation of the system. Formal methods based technique, i.e., VDM-SL is used to develop formal representation of the proposed model. The proposed model is validated, verified and analyzed using VDM-SL Toolbox.

The approach we have used can be applied to implement any algorithm in wireless sensor and actor networks. This is because we have observed that there is an analogous relationship between graph theory models and VDM-SL structures. By integrating graph theory with VDM-SL, it has become possible for modeling large and complex software systems using integration of approaches.

REFERENCES

[1] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz, "BorderSense: Border patrol through advanced wireless sensor networks," Ad Hoc Networks, Vol. 9, No. 3, pp. 468-477, 2011.

[2] C. Komar, M. Y. Donmez, and C. Ersoy, "Detection quality of border surveillance wireless sensor networks in the existence of trespassers' favorite paths," Computer Communications, Vol. 35, No. 10, pp. 1185-1199, 2012.

[3] M. Alkhathami, L. Alazzawi, and A. Elkateeb, "Border surveillance and intrusion detection using wireless sensor networks," International Jouranl of Advances in Engineering & Technology, 2015.

[4] N. A. Zafar, "Modeling and formal specification of automated train control system using Z notation", IEEE Multitopic Conference (INMIC'06), pp. 438-443, 2006.

[5] N. A. Zafar., "Formal dynamic operational model of RIS components," IJCSNS, Vol. 11, No. 9, p. 91, 2011.

[6] S. Yousaf, N. A. Zafar, and S. A. Khan, "Formal analysis of departure procedure of air traffic control system," IEEE 2nd International Conference on Software Technology and Engineering (ICSTE), Vol. 2, pp. V2-301, 2010.

[7] M. Conrad, and D. Hötzer, "Selective integration of formal methods in the development of electronic control units," Proceedings of IEEE 2nd International Conference on Formal Engineering Methods, 1998.

[8] H. Afzaal, and N.A. Zafar, "Formal modeling and algorithm of subnet-based backup assigning in WSAN," 6th International Conference on Information and Communication Technologies (ICICT), 2015.

[9] C. C. Haddal, and J. Gertler, "Homeland security: unmanned aerial vehicles and border surveillance," Library of Congress Washington DC Congressional Research Service, 2010.

[10] S. Riaz, H. Afzaal, M. Imran, N. A. Zafar, and M. S. Aksoy, "Formalizing mobile ad hoc and sensor networks using VDM-SL," Procedia Computer Science, Vol. 63, pp. 148-153, 2015.

[11] H. Afzaal, M. Imran, and N. A. Zafar, "Implementing partitioning detection and connectivity restoration in WSAN using VDM-SL," Frontiers of Information Technology, 2015.

[12] M. Imran, N. A. Zafar, M. A. Alnuem, M. S. Aksoy, and A. V. Vasilakos, "Formal verification and validation of a movement control actor relocation algorithm for safety–critical applications," Wireless Networks, pp. 1-19, 2015.

[13] M. Imran, and N.A. Zafar, "Formal specification and validation of a hybrid connectivity restoration algorithm for wireless sensor and actor networks," Sensors, Vol. 12, No. 9, pp. 11754-81, 2012.

[14] M. Alnuem, N. A. Zafar, M. Imran, S. Ullah, and M. Fayed, "Formal specification and validation of a localized algorithm for segregation of critical/noncritical nodes in MAHSNs," International Journal of Distributed Sensor Networks, 2014.