Afzaal and Zafar *Complex Adapt Syst Model (2016) 4:27*
DOI 10.1186/s40294-016-0037-4

Complex Adaptive Systems Modeling

RESEARCH Open Access

Formal analysis of subnet-based failure recovery algorithm in wireless sensor and actor and network

Hamra Afzaal and Nazir Ahmad Zafar*

*Correspondence:
nazafar@ciitsahiwal.edu.pk
Department of Computer Science, COMSATS Institute of Information Technology, Sahiwal Campus, Sahiwal 57000, Pakistan

Abstract

Wireless sensor and actor networks (WSANs) have various applications in safety and mission critical systems. Sensors are used for sensing the information whereas actors for taking intelligent decisions. Developing and modeling algorithms for WSANs have raised several research issues which have captured attention of the research community. Maintaining inter-actor connectivity or failure recovery is a critical issue in WSANs because these are deployed in harsh and inhospitable environment which may result into physical damage to actors loosing inter-actor connectivity. In case of failure of inter-actor connectivity, the topology of the network may be affected that might be inefficient to recover. Therefore an efficient subnet-based failure recovery algorithm (SFRA) is proposed in this work. It is assumed the partitioning of WSAN into subnets which localizes the failure recovery procedure at subnet level achieving objective of efficiency. Moreover, algorithm is hybrid as it assumes pre-failure planning and post-failure recovery. The proposed model is presented as a graph-based model to represent static part of the network topology. The graph model is transformed into a formal model using Vienna development method-specification language (VDM-SL). The static model is described by defining formal specification of subnets, network topology, sensors, actors and gateways as composite objects. The state space of the WSANs is described in the form of functions and operations as dynamic part of the model. Invariants are defined over the data types in static model for ensuring safety criteria and pre/post conditions are defined in functions and operations for changing state space of the system. The proposed model is validated and verified using VDM-SL Toolbox.

Keywords: Wireless sensor and actor networks, Failure recovery, Modeling, Graph theory, Formal specification, VDM-SL, Validation and verification

Background

Wireless sensor and actor networks (WSANs) have various applications in safety and mission critical systems because of their suitability in remote and harsh areas. Body area networks (Fortino et al. 2015), building management systems (Fortino et al. 2012), internet of things (Fortino and Trunfio 2014) and WSN invulnerability (Fu et al. 2013) are few important application areas of WSAN.

WSANs are complex adaptive systems (CAS) because of using complex adaptive environment. WSANs employ sensors for complex sensing and actors for taking intelligent

decisions and appropriate actions. Sensors have low cost, low battery power, less processing capability and short range of communication while actors are more expensive having powerful battery, processing capability and long range of communication. Developing and modelling of algorithms for WSANs have raised various research challenges. For example:

- Deploying and maintaining inter-actor connectivity and failure recovery of actors for the responsiveness of the entire network.
- Developing solutions and algorithms for energy efficiency of network communications due to resource-constrained sensors and actors.
- Addressing communication and coordination problems in real-time due to co-existence of sensors and actors to guarantee the timely execution of correct actions.
- Ensuring safety and security of the data communicated among sensors and actors.

In this paper, it is focussed on the first two research challenges by developing a localized, energy efficient and hybrid algorithm for the failure recovery. Graph-based formal model is developed to describe the algorithm at detailed-level for verification. A graph consists of set of vertices, V, and set of edges, E, which is denoted by G = (V, E) to represent topology of the network. The edge-set E is assumed as a distinct unordered pairs of distinct elements of a set V. A subnet of the network having vertex set V and edge set E is a network having vertex-set V′ and edge-set E′ contained in V and E respectively. Graphical representation of the topology of WSANs is presented as in Fig. 1.

Graph theory and networks have analogous nature, for example, vertices of graph represent sensors or actors in a network and edges in graph represent wireless communication links in the network. That is why graph theory is used as a semi-formal way of storing and processing the information in our proposed algorithm. Moreover, if the semi-formal model is developed based on graph theory then it is easier to transform it to a formal model by defining a mapping among both the approaches. In most of the existing

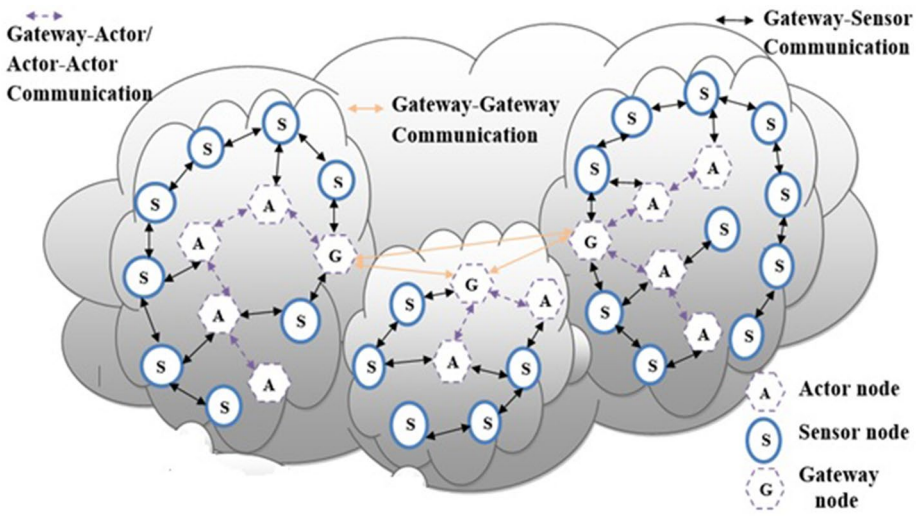

Fig. 1 Subnets-based representation of WSANs

work, failure recovery algorithms of WSANs are validated through testing or simulation techniques. Due to certain limitations of simulation, formal methods are employed in our work to describe the proposed algorithm for failure recovery in WSANs.

This paper is continuation of our earlier work in this area (Afzaal et al. 2015; Imran and Zafar 2012; Afzaal and Zafar 2015a, b, c, 2016). In this paper, subnet-based failure recovery algorithm (SFRA) is developed and formalized which assumes partitioning of WSAN into subnets localizing the problem and increasing energy efficiency of the recovery process. A subnet having fixed size p is defined as a collection of n number of sensors and m number of actors. One gateway node which is actually an actor node is assumed in a subnet. The subnets are connected through gateway nodes. The total number of subnets depends on the total number of nodes and size of the subnet which may depend upon the application. The partitioning of WSAN into subnets increases the life time of the network because the failure recovery procedure is localized in a subnet. The subnet based approach is different from clustering as follows:

- In subnet based approach, it is assumed that sensors and actors are deployed randomly in the form of subnets saving energy. In clustering approach, sensors and actors are deployed to form clusters formulating groups together consuming energy.
- In subnet based approach, it is assumed that subnets are disjoint while in the cluster based approach there may be overlapping of nodes, that is, a node in a cluster may serve to more than one clusters depleting the energy more quickly.
- In subnet based approach, the gateway node receives information of the subnet while in clustering approach the cluster head receives information of the cluster. The gateway node being critical actor is assigned backup which serves in case of failure of the gateway while the cluster head is reselected randomly when its energy is depleted. It is noted that assigning backup to the gateway requires less energy as compared to reselection of a cluster head.
- In subnet based approach, it is assumed that if an event is sensed by a node it is not sensed by the other node removing redundant information from the network.

The SFRA is hybrid as it assumes pre-failure planning and post-failure recovery of an actor node. In the algorithm, the backup assigning procedure is defined as a reusable function increasing robustness of the formal model. SFRA is an integration of centralized and distributed approach because the control is distributed among actors in a subnet and centralized among the gateway nodes. A gateway node receives information of a subnet and communicates with other gateways of the subnets. In the centralized approach, it is assumed the existence of a central object that may not be part of the subnets. This approach is useful in static environment but increases the computational cost in terms of energy and time. On the other hand, the distributed approach is dynamic and addresses the unpredictable environments but is not efficient due to lack of complete information.

The detailed formal model of SFRA is described using Vienna development methods-specification language (VDM-SL) because of its expressive power of detailed description and having rigorous computer tool support (SCSK Corporation 2013). WSANs are modelled as dynamic graph because actors are mobile and the topology may change most frequently. Sensors and actors are defined using composite objects which may

have more than one type. Invariants are defined over composite object types to describe their safe behaviour. Pre and post conditions are defined in functions and operations to verify the consistency and correctness of behaviour. The results are visualized through the validation techniques available in the VDM-SL toolbox (SCSK Corporation 2013) which increase the confidence of correctness. The rest of the paper is organized is as follows: The next subsection illustrates an introduction to complex adaptive systems. System model, problem statement and the proposed algorithm are presented in "Methods" section. Formal specification and analysis of the algorithm is described in "Results and discussion" section. In "Related work" section, the related work is discussed critically. Conclusion and future work are discussed in "Conclusions" section.

Modelling complex adaptive systems

CAS are defined as dynamic networks of a large number of agents, individuals, species, cells, nations, firms that act in parallel, act and react constantly to what the other agents do (Zafar 2016; Holland 2006). Examples of artificial CAS include artificial intelligence systems, evolutionary programs and artificial neural networks. The control of CAS is required to be coherent as it is greatly centralized and decentralized (Armano and Javarone 2013). In CAS, many decisions are made by large number of agents which influence each other, interact with each other, change their behaviours and learn from their experiences to achieve the overall behaviour of a system. CAS analysis can be done using experimental, applied and theoretical methods like computer-based simulation and mathematical modeling.

CAS are complex in nature and requires rigorous tool support. These tools are complex themselves to develop and understand such systems. CAS are analyzed through computer simulation tools using agent based methodologies (ABM) and complex networks (CN). ABM simulates CAS at detailed level which represents interactions and actions of different agents in the artificial world. Agent-based models are the models which represent recursive mathematical functions, computer code that are applied to a definite set of inputs. These models can present explicitly the patterns of agent's behaviour, micro interactions and are not limited to drive equations of a system or representing general statistical models (Boulaire et al. 2015). Mathematical and statistical analysis techniques still have importance in playing critical role in developing and testing the ABM. CN are effective to define CAS like neural networks, chemical systems, biological systems and the World Wide Web. Initially these types of systems are defined by graphs where nodes represent agents, and edges represent relationship between the agents. CN support in defining the structural properties of CAS by describing topology of the system and then describing rules that govern agents behaviour. Many questions arise while studying and applying complex networks, e.g., how to model large and complex networks having complex topology and behave collectively.

For the verification of CAS (Zafar 2016), most of the researchers focused on ABM and CN modeling which are based on testing and simulation techniques. These techniques cannot guarantee about the correctness of the systems because to gain a required level of confidence number of inputs for testing and simulations increases exponentially. Moreover, if it is required to enhance the system then regression testing will be required to perform which necessitates that a complete set of simulations must be re-performed.

Formal approaches are effective to prove correctness of the models and are reliable to overcome the limitations of testing and simulation (North 2014; Bouarfa et al. 2013; Shah et al. 2015). In addition, formal specification helps to perform better simulations and provide exhaustive support for the verification of algorithms that is why these are required to apply before the simulation.

Methods

System model and problem statement

Subnet-based failure recovery algorithm (SFRA) is developed by partitioning WSAN into subnets. The advantage of partitioning is that the problem is localized and energy efficiency is increased because a subnet requires less memory as well as less computation. A subnet employs sensors, actors and gateways which are in fact the most powerful actors. Actors are powerful in terms of battery, processing and wireless communication than sensors. An actor is assumed to have long communication range than a sensor. Sensors and actors are deployed in an area of interest in the form of subnets. Actors discover each other and form inter-actor connectivity in a subnet. Most powerful actor is selected as a gateway in a subnet for communication with the other gateways of the subnets. Short range communication interface is used for communication of sensors and actors within a subnet and a long range interface is used for communication of gateways among the subnets.

A scenario of connectivity of subnets is shown in Fig. 2. The effect of an actor failure in a subnet depends on its position in the subnet. In a network, there are two types of actors, i.e., critical or non-critical. Leaf of a subnet is assumed as non-critical. Failure of a non-critical actor, for example S_{12} in the Figure, does not affect the connectivity of a subnet while failure of a critical actor, S_{13} in the Figure, divides the subnet into disjoint segments. Moreover failure of a gateway node, for example S_{11}, disconnects the subnet with other subnets. To handle critical node failure, three type of approaches are used, i.e., proactive, reactive and hybrid. The proactive approaches focus on establishing

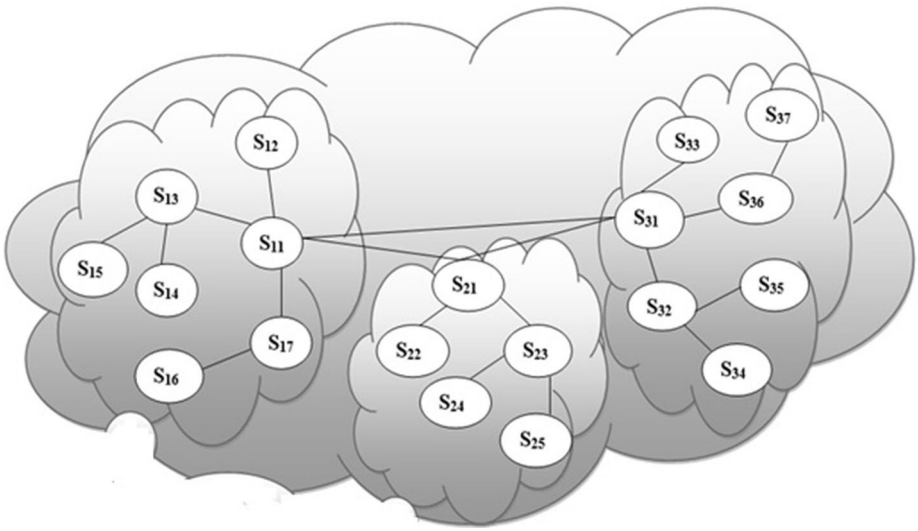

Fig. 2 Scenario of connectivity of subnets

and maintaining bi-connected topology which leads to large actor count and higher cost which is impractical. In reactive approaches, the recovery process is initiated when the failure is detected. Hybrid approaches are more suitable for complex, mission critical and safety–critical applications because of pre-failure planning and post-failure recovery. Our proposed algorithm is hybrid which better suits for WSANs as failure recovery time in a subnet is minimized because of local recovery.

Subnet-based failure recovery algorithm

For safety and mission critical, and time sensitive applications hybrid algorithms suit better because these applications require rapid recovery process. Our proposed SFRA algorithm is hybrid as it consists of two parts, i.e., pre-failure planning and post-failure recovery. In the pre-failure planning, SFRA identifies a gateway and critical actors in a subnet and designates them appropriate backups in a subnet. The gateway is a critical actor therefore it is also assigned a backup. A gateway actor is used for communication with the other subnet gateways. The backup monitors critical or gateway actor and detects the failure through missing the heartbeats. In post-failure recovery, the backup of a critical actor moves to the location of the critical actor and the process takes place in a cascaded manner. As the gateway is also a critical actor therefore post-failure recovery procedure is same as for the critical actor. The detailed algorithm is described in the next.

Partitioning into subnets

In our algorithm, we assume WSAN partitioned into connected subnets as shown in Fig. 2. Our network and subnets are analogous to graph and sub-graphs respectively in graph theory. There exist various algorithms to partition a graph into almost equal number of sub-graphs (Borozan et al. 2016). It is noted that how to partition the network into subnets is not considered in this paper. A subnet consists of sensor and actor nodes. One of the actor nodes of a subnet is designated as the gateway node. Sensor nodes detect events from the environment and report to nearby actor nodes. Actor nodes coordinate with each other for the optimal response. In this way, a problem is solved locally in a subnet. Moreover, the cascading process is reduced at subnet level which conserves energy and reduces the time complexity of the algorithm. The subnets are connected through gateway nodes which are responsible for communication with the other subnets gateway nodes. There should be at least two nodes in a subnet, i.e., one is a gateway node and other is its backup. There should not be any repeated nodes in any subnet. All nodes in a subnet communicate using short range communication interface.

Pre-failure planning

Failure of a gateway node disconnects the subnets and failure of a critical actor in a subnet disconnects each of its neighbors. The neighbors become unable to communicate because of loss of connectivity. Therefore SFRA pursues pre-failure planning to identify gateway and critical nodes and designate the appropriate backups. The gateway actors have most of the information of the subnets and are responsible for connecting and communicating among the subnets. It is supposed that a gateway node in a subnet

communicates with the other gateways of the subnets using long range communication interface. The gateway node is selected based on the power, degree and position. The gateway node should have highest power as the most powerful gateway nodes have the more ability to sustain. SFRA prefers the actor to be selected as a gateway that has highest degree. That means the strongly connected nodes with its neighbors are more appropriate to perform as gateway nodes because such nodes have most of the information of the subnets to communicate with the other subnets. If multiple actors have the same power and degree then the node that is more close to the other nodes is selected as a gateway which will shorten the communication time which is crucial in critical applications.

Failure of a critical actor in a subnet divides the subnet into disjoint segments. The loss of a non-critical actor or leaf node from the subnet does not affect the connectivity of a subnet. SFRA requires 1-hop positional information to identify critical actors in each subnet and designate appropriate backup for the critical actors. The process of critical actor identification runs in every subnet. It reduces computation overhead in a subnet as less processing is needed as compared to processing required in the entire network. Once the gateway actors are selected and critical actors are identified, they are assigned appropriate backups among the neighbors. Depending on the application area, several criteria can be defined when choosing a backup. It is supposed that a gateway cannot be a backup of any critical node. In our proposed algorithm, the backup is selected among 1-hop neighbours based on power, non-critical neighbour, actor degree and position. The most powerful actor among the neighbors of a critical node is selected to be assigned as the backup as it has more ability to sustain. Non-critical neighbour which is an actor is preferred to be assigned as a backup for the critical actor. This is because the movement of such nodes will not affect the inter-actor connectivity of the overall subnet. Moreover, it restricts the scope of recovery and reduces the movement overhead of the nodes of the subnet. There is a significant movement cost and impact on moving a node that has many neighbors. If a non-critical node is not available among the neighbors, SFRA prefers to replace the failed actor with a neighbor which is critical actor but has a highest degree in the competitors. That means it is strongly connected with the neighbors because there is more probability to have non-critical nodes in the neighbor. In this way, it reduces the recovery overhead and restricts scope of cascaded relocation. If multiple critical neighbors have the same power and actor degree then a nearby neighbor is preferred to be assigned as a backup. It will reduce the movement overhead and shorten the recovery time that is crucial and required for resource-constrained and mission-critical applications. By following the above procedure, backups are selected for all critical nodes and a gateway node in each subnet. An actor may serve as a backup for many critical nodes. If the backup actor fails then the critical actor selects another actor as a backup using the same procedure as stated above.

Post-failure recovery

After the gateway nodes and critical actors are identified, the backups are assigned as discussed above. For example, S11, S13 and S17 are critical nodes and are assigned backups, S12, S14 and S16 respectively as in Fig. 3a. The backups are notified through regular

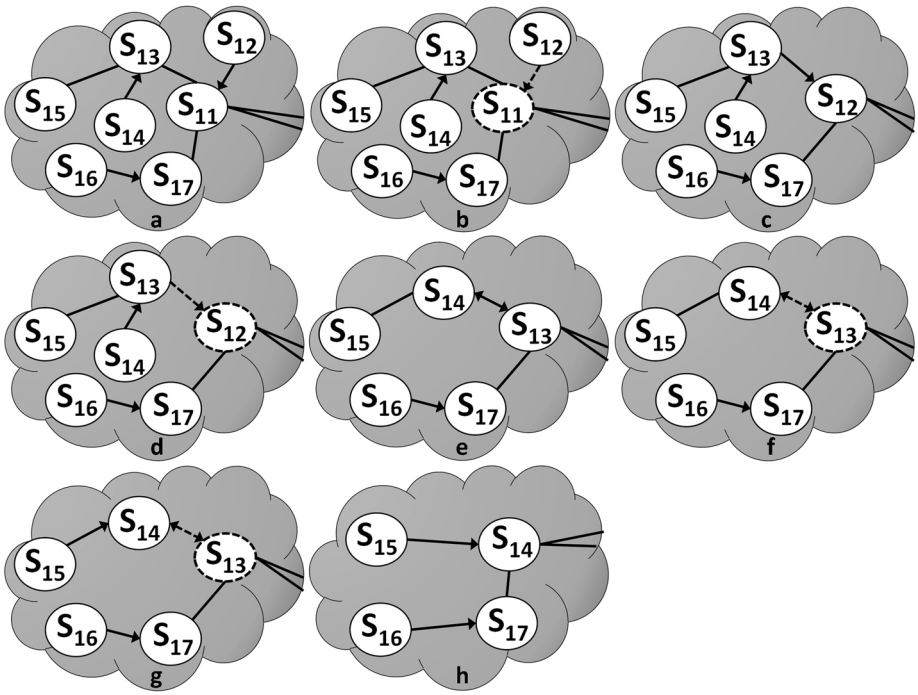

Fig. 3 Recovery process of SFRA

heartbeat messages of critical actors and gateway nodes. The pre-designated backup starts monitoring its primary (gateway or critical) through periodical heartbeats. The backup declares failure of its primary after continuous missing of the heartbeats. After failure detection of a primary, the backup triggers the post-failure recovery procedure. There are three different scenarios: (1) backup is non-critical, (2) backup is critical and has its primary as backup and (3) backup is critical and its backup is other than its primary. As the node S12 is the backup of a primary critical actor S11. It is noted that the node S12 detects failure of its primary S11 as in Fig. 3b. The backup S12 is a non-critical actor and it simply replaces its primary S11 in case of its failure by first scenario as in Fig. 3c. The node S12 has become critical at this place that is why it identifies a node S13 to be designated as a backup. The nodes S12 and S13 are critical nodes and backup of each other as in Fig. 3d (scenario 2), hence, a cascaded relocation will be performed in case of failure of any one. The backup S13 of the primary S12 triggers the recovery procedure once it detects the failure of S12 as in Fig. 3e. The actor S13 detects failure of S12 and selects another actor S14 as a backup. The backup S13 moves to the position of S12 as in Fig. 3f. The newly assigned backup actor S14 of S13 performs a cascaded relocation and so on. The third scenario is when the backup is a critical node and has its backup as other than the primary node. For example, the nodes S13 and S14 are critical nodes as in Fig. 3g. The node S14 is the backup of S13 and S15 is the backup of S14. The actor S14 detects failure of S13 and moves to the position of S13 as in Fig. 3h. The backup S15 of S14 replaces the S14 by cascading procedure and so on by completing the recovery procedure. Every time a backup is assigned to a critical node, it notifies its own backup so that the network stays

connected. It is mentioned that this scenario may trigger a series of cascaded repositioning of nodes. In case of failure of the backup, there are three possible scenarios. Firstly, if the backup of backup is non-critical then it replaces its primary actor and further relocation is not required. The other two scenarios work in the similar way as described above.

Pseudo code of SFRA

A high level pseudo code of the SFRA is shown in Fig. 4 running on the topology of the entire network in a distributed manner. The set of all the notations and functions used in the algorithm are listed and explained before the pseudo code. At first, all the sensors and actors are deployed randomly in the form of subnets (line 1). The gateway nodes are selected based on the criteria given above (line 2). Initially, all the actors of all the subnets are initialized as non-critical (lines 3–5). For every actor A of every subnet SN, a localized cut-vertex detection procedure determines whether the given node A is critical or not (lines 6–8). If the actor A is critical then an appropriate backup actor B among the neighbors is selected (lines 9–11). Upon detecting failure of the primary A, the backup initiates the recovery procedure. If the backup actor B is non-critical then it simply moves to the position of A and as it becomes critical at that place it identifies a node to be designated as a backup (lines 12–14, 34–36). If backup node B is critical and simultaneously primary and backup of critical node A then it selects another node as backup. In this way, the recovery process is completed by notifying and cascading, that is, backup B of A is moved to position of A and newly backup of B is moved to position of B (lines 15–17, 37–42). If backup node B is critical and its backup is other than its primary then it notifies to its back and moves to the position of its primary by cascading. Finally, the backup of B is moved to the position of B (lines 18–21, 43–46). If the backup is failed then the same procedure is called recursively for the failure recovery of the subnet (lines 22–33). The notations used in the algorithm are listed below.

- T = Topology(A, S, G, SN, N, L)
- A = Set of all actors = {A1, A2,…, Am}
- S = Set of all sensors = {S1, S2,…, Sn}
- Maximum size of a subnet = p
- Total number of nodes = N= m + n
- Number of subnets = K= (m + n)/p
- G = Set of all gateways = {G1, G2,…, GK} $\subseteq$ {A1, A2,…, Am}
- SN = Set of all subnets = {SN1, SN2,…, SNK}
- L = Set of all possible links among actors, sensors and gateways
- IsCritical (A) == returns true if A is critical otherwise false
- Neighbors(A) == returns set of all neighbors of A
- BackupOf (A) == returns backup for A.
- NotEqual(B, A) == TRUE if B is not equal to A
- AssignBackup(B, A) == B is assigned as backup to A
- Move(B, A) == Move B to location of A
- BothBackups(B, A) == TRUE if both B and A are primary and backups of each other
- NotifiesToBackup(A) == A notifies to its newly assigned backup
- ReplaceBackup(A) == Returns newly backup of A

```
SFRA (T): T is a topology
1. Deploy the nodes, i.e., actors, sensors and gateways randomly formulating
   subnets
2. Select a gateway Gi for every subnet SNi, for i = 1, 2, . . ., k
3. FORALL subnet SNi
4.    FORALL actors A in set SNi
5.    IsCritical(A) ← FALSE
6. FORALL A in subnet SNi, IF any of the node in Neighbors(A) is
   disconnected without A THEN
7.    IsCritical(A) ← TRUE
8. ENDIF
9. IF IsCritical(A) ← TRUE THEN
10.   AssignBackup(BackupOf(A), A)
11.ENDIF
12.IF (Primary actor A fails) THEN
13.   IF (IsCritical(BackupOf(A)) == FALSE) THEN
14.      FR-1(A)
15.   ELSE
16.   IF BothBackups(BackupOf(A), A) THEN
17.      FR-2(A)
18.   ELSE
19.      FR-3(A)
20.   ENDIF
21.   ENDIF
22.  IF (Primary actor BackOf(A) fails) THEN
23.     IF (IsCritical(BackupOf(BackOf(A))) == FALSE) THEN
24.        FR-1(BackOf(A))
25.     ELSE
26.     IF BothBackups(BackupOf(BackOf(A)), BackOf(A)) THEN
27.        FR-2(BackOf(A))
28.     ELSE
29.        FR-3(BackOf(A))
30.     ENDIF
31.   ENDIF
32.   ENDIF
33.ENDIF
34.FR-1(A)
35.   Move(BackupOf(A), A)
36.   AssignBackup(IdentifyBackup(BackupOf(A)), BackupOf(A))
37.FR-2(A)
38.   ReplaceBackup(BackupOf(A))
39.   AssignBackup(ReplaceBackup(BackupOf(A)), BackupOf(A))
40.   NotifiesToBackup(ReplaceBackup(BackupOf(A)))
41.   Move(BackupOf(A), A)
42.   Move(ReplaceBackup(BackupOf(A)), BackupOf(A))
43.FR-3(A)
44.   NotifiesToBackup(BackupOf(A))
45.   Move(BackupOf(A), A)
46.   Move(BackupOf(BackupOf(A)), BackupOf(A))
```

Fig. 4 High-level pseudo code of SFRA

The time complexity of deploying nodes and selecting gateways of the subnets (line 1–2) is N. Time complexity of initializing all the actors as non-critical (lines 3–5) is KN. Time required for checking criticality and assigning backups to actors (lines 6–11) is N. If the backup is non-critical then the recovery procedure (lines 12–14, 34–36) takes time as KN. If the backup is non-critical and simultaneously primary then the recovery procedure (lines 15–17, 37–42) takes time as N^K. If the backup is non-critical and simultaneously primary and backup of a critical node then the recovery procedure (lines 18–21, 43–46) also takes time as N^K. If the backup is failed then recursive failure procedure

takes (lines 22–33) time as N^K. Hence the maximum time complexity of the algorithm in terms of Big O is N^K. As the size of subnet k is a fix number which does not depend upon the number of nodes hence the time complexity of the algorithm is polynomial type.

Results and discussion

Formal model using VDM-SL

This section presents formal specification of the proposed algorithm for WSANs. Data types are defined for the static modelling whereas state, functions and operations are defined for the dynamic modelling. Several constructs, for example, sets, sequences, composite objects, maplets, invariants, pre and post conditions, are used for well defining the specification. VDM-SL Toolbox is used for the analysis of the model.

Static model

Firstly, static model of the proposed algorithm for WSANs is presented. The WSANs consists of sensor, actor and gateway nodes which have some common characteristics and are defined by the composite object *AbstractObject*. It is defined by six fields, namely, *node*, *pwr*, *states*, *information*, *neighbors*, *connectivity*. The description of its fields is given in Table 1.

```
types

Position::xc:int
          yc:int;

Node::sensor:Sensor
      actor:Actor
      gateway:Gateway
      position:Position;

Pwr = <<HIGH>>|<<LOW>>; State = <<SENSED>>|<<NOT_SENSED>>;
Data = token; Connectivity = <<CONNECTED>>|<<DISCONNECTED>>;

AbstractObject::node:Node
                 pwr:Pwr
                states:State
             information:set of Data
             neighbors: set of Neighbor
            connectivity:Connectivity;
```

Table 1 Fields of abstract object

#	Field name	Field description
1	*Node*	A node may be a sensor, actor or gateway having certain position. Position of a node is recorded in the form of x and y-coordinates
2	*Pwr*	The nodes have some power which is represented as union type, i.e., HIGH or LOW
3	*States*	It records status of sensor, actor and gateway represented as union of quote types
4	*Information*	This field is used for recording the sensed data
5	*Neighbours*	It illustrates that the neighbor nodes exists in the connected network
6	*Connectivity*	It checks the connectivity status of a sensor, actor and a gateway

We assume WSAN partitioned into connected subnets. In a subnet, sensors sense events from the environment and transmit the information to nearby actors which coordinate with each other for the optimal response. A gateway in a subnet has information of a subnet and communicates with other subnets. A subnet is defined by composite object *Subnet* having three fields, namely, *nodes*, *edges* and *communication_interface*. The description of its fields is shown in Table 2.

```
Edge = Node * Node
inv edge == let mk_(n1, n2) = edge in  n1 <> n2;

Edges = set of Edge
inv edges == forall mk_(n1, n2) in set edges & mk_(n2, n1) in set edges;
Communication_Interface=<<BLUETOOTH>>|<<WiFi>>;
Communication_Range=<<SHORT>>|<<LONG>>;
Comunication_Interface::ci:Communication_Interface
                          communication_range: Communication_Range;

Subnet::nodes: set of Node
        edges:Edges
         communication_interface:Comunication_Interface
inv mk_Subnet(nodes,edges, communication_interface)== card nodes >=2 and forall n1, n2 in set nodes &
   exists e in set edges & mk_(n1,n2) = e and forall e in set edges & exists n1, n2 in set nodes
   & mk_(n1,n2) = e and communication_interface.ci=<<BLUETOOTH>>
  and   communication_interface.communication_range=<<SHORT>>;
```

Invariants (1) Every subnet must employ at least two nodes. (2) Any two nodes are connected by an edge in the subnet. (3) All the edges in the subnet consist of two nodes. The edges describe communication in the subnet. Any isolated node does not exist in the subnet. (4) The communication interface within the subnet is assumed as Bluetooth and its communication range is short.

All the subnets in the network collectively define topology of WSAN. In the specification, it is defined by a composite object *NetworkTopology* which consists of two fields, namely, *subnets* and *edges*. In the invariant, it is stated that for each subnet, there exist a gateway node connected to the other gateway nodes of subnets. The description of fields of network topology is described in Table 3.

Table 2 Description of fields of a subnet

#	Field name	Field description
1	*Nodes*	The first field, *nodes*, is a collection of nodes in a subnet
2	*Edges*	Edges represent communication links between nodes
3	*Communication_interface*	This field is used for describing communication within the subnet

Table 3 Fields of network topology

#	Field name	Field description
1	*Subnets*	This field represents a set of subnets in the network topology
2	*Edges*	The field *edges* is used to describe communication among the subnets

NetworkTopology::*subnets*:**set of** *Subnet*
edges:**set of** *Edge*
inv mk_*NetworkTopology*(*subnets*, *edges*)== **forall** *s* **in set** *subnets* & (**exists** *n* **in set** *s.nodes* &
(**exists** *g1*,*g2* **in set** {*n.gateway*} & (**exists** *e* **in set** *edges* & **mk**_(*g1.gateway_id*, *g2.gateway_id*)=*e*))) ;

As the network topology consists of sensor, actor and gateway nodes in the subnets. Sensor, actor and gateway are specified as composite objects in the formal specification. The composite object *Sensor* is defined by three fields, namely, *sensor_id*, *sensor_fields* and *position*. The description of fields of sensor is presented in Table 4.

In the invariants, it is stated that: (1) A sensor node has low power. (2) A sensor sensing state is sensed if and only if it stores some sensed information. (3) A sensor is connected if and only if it has some neighbours.

An actor is specified as a composite object *Actor* having six fields namely, *actor_id*, *actor_type*, *backup*, *actor_fields*, *action* and *position*. The description of the fields of actor is illustrated below in Table 5.

Sensor::*sensor_id*:*Node*
sensor_fields:*AbstractObject*
position:*Position*
inv mk_*Sensor*(-,*sensor_fields*,-)== *sensor_fields.pwr*=<<LOW>> **and** *sensor_fields.states*=<<SENSED>>
<=> *sensor_fields.information*<>{} **and** *sensor_fields.states*=<<NOT_SENSED>> <=>
sensor_fields.information={} **and** *sensor_fields.connectivity*=<<CONNECTED>> <=>
sensor_fields.neighbors <>{} **and**
sensor_fields.connectivity=<<DISCONNECTED>> <=> *sensor_fields.neighbors*={};

Table 4 Description of fields of sensor

#	Field name	Field description
1	*Sensor_id*	Every sensor node is unique
2	*Sensor_fields*	This field is used to access common fields from the abstract object
3	*Position*	This field is used to record the position

Table 5 Description of fields of actor

#	Field name	Field description
1	*Actor_id*	Every actor node is unique
2	*Actor_type*	It records criticality of an actor
3	*Backup*	Backup is required for a critical actor
4	*Actor_fields*	This field is used to access common fields from the abstract object
5	*Action*	This field represents action performed by an actor after receiving any information
6	*Position*	It is required to record the position of an actor node in the network

Criticality = <<CRITICAL>>|<<NONCRITICAL>>; *Action* = **token**; *Location* = **token**;

Actor::*actor_id*:*Node*
actor_type:*Criticality*
backup:*Neighbor*
actor_fields:*AbstractObject*
action:*Action*
position:*Position*

inv mk_*Actor*(*actor_id*, *actor_type*, *backup*, *actor_fields*,-,-)== *actor_fields.pwr*=<<HIGH>> **and**
actor_type=<<CRITICAL>> => **card** *actor_fields.neighbors* <= 2 **and**
actor_type =<<NONCRITICAL>>=> **card** *actor_fields.neighbors* > 2 **and**
exists *nr* **in set** *actor_fields.neighbors* & (*backup* =*nr*) **and**
backup.neighbor_id=*actor_id* **and** *backup.pwr*=<<HIGH>> **and**
backup.criticality=<<NONCRITICAL>> **or** *backup.criticality* <> <<NONCRITICAL>>
=> *backup.criticality*=<<CRITICAL>>**and forall** *nr* **in set** *actor_fields.neighbors* &
(**card** *backup.neighbors*>= **card** *nr.neighbors*) **and** (**abs** (*actor_id.position.xc*-*backup.position.xc*) +
abs (*actor_id.position.yc*-*backup.position.yc*)) <= (**abs** (*actor_id.position.xc*-*nr.position.xc*) +
abs (*actor_id.position.yc*-*nr.position.yc*)) **and** *actor_fields.states*=<<SENSED>>
<=>*actor_fields.information*<>{} **and** *actor_fields.states*=<<NOT_SENSED>>
<=>*actor_fields.information*={} **and** *actor_fields.connectivity*=<<CONNECTED>>
<=> *actor_fields.neighbors* <>{} **and** *actor_fields.connectivity*=<<DISCONNECTED>>
<=> *actor_fields.neighbors*={};

Invariants (1) The actor must have high power as compared to sensor. (2) Actor is critical if it has two or less than two neighbours otherwise is non-critical. (3) The backup of the actor is selected from its neighbours which must be an actor. (4) The backup should have high power and non-critical backup should be preferred. (5) If non-critical backup is not available among the neighbor nodes then critical backup is selected. (6) The backup of the actor is the one with highest degree among its neighbors. (7) The backup of the actor should be nearly positioned as compared to the other neighbours. (8) An actor sensing state is sensed properly. (9) An actor is connected if and only if it has some neighbours.

Sensors and actors communicate the information with a gateway in a subnet. The formal specification of a gateway is presented as a composite object *Gateway* having seven fields, namely, *gateway_id*, *actor*, *gateway_type*, *backup*, *gateway_fields*, *communication_interface* and *position*. The description of fields of gateway is specified below in Table 6.

Table 6 Description of fields of gateway

#	Field name	Field description
1	*Gateway_id*	Every gateway node is unique
2	*Actor*	The gateway node has all the characteristics of an actor node
3	*Gateway_type*	It is required for representing criticality of a gateway
4	*Backup*	This field shows that a backup is selected from the neighbors
5	*Gateway_fields*	It is used to access common fields from the abstract object
6	*Communication_interface*	A gateway node communicates using communication interface
7	*Position*	It is required to record the position of a gateway node in the network

Gateway::*gateway_id*:*Node*
actor:*Actor*
gateway_type:*Criticality*
backup:*Neighbor*
gateway_fields:*AbstractObject*
communication_interface:*Comunication_Interface*
position:*Position*
inv mk_*Gateway*(*gateway_id*, *actor*, *gateway_type*, *backup*, *gateway_fields*,
communication_interface,-)==*gateway_id*=*actor.actor_id* **and** *gateway_type*=<<CRITICAL>> **and**
forall *a* **in set** {*actor*} & (*a.actor_fields.pwr*=<<HIGH>>=>*gateway_id*=*a.actor_id*) **and**
exists *a1* **in set** {*actor*}& (*a*<>*a1* **and card** *a.actor_fields.neighbors*>= **card** *a1.actor_fields.neighbors*
=> *gateway_id*=*a.actor_id*) **and exists** *nr* **in set** *gateway_fields.neighbors* &
(*backup*=*nr* **and** *backup.pwr*=<<HIGH>> **and** *backup.criticality*=<<NONCRITICAL>>
or *backup.criticality* <> <<NONCRITICAL>> => *backup.criticality*=<<CRITICAL>> **and**
card *backup.neighbors* >= **card** *nr.neighbors* **or** (**abs** (*gateway_id.position.xc*-*backup.position.xc*) +
abs (*gateway_id.position.yc*-*backup.position.yc*)) <= (**abs** (*gateway_id.position.xc*-*nr.position.xc*) +
abs (*gateway_id.position.yc*-*nr.position.yc*))) **and** *gateway_fields.states*=<<SENSED>>
<=> *gateway_fields.information*<>{} **and** *gateway_fields.states*=<<NOT_SENSED>>
<=> *gateway_fields.information*={} **and** *gateway_fields.connectivity*=<<CONNECTED>>
<=> *gateway_fields.neighbors* <>{} **and**
gateway_fields.connectivity=<<DISCONNECTED>> <=> *gateway_fields.neighbors*={} **and**
communication_interface.ci=<<WiFi>> **and** *communication_interface.communication_range*=<<LONG>> ;

Invariants (1) The gateway node must be an actor node which is critical. (2) The actor node that has high power and is strongly connected is selected as a gateway node. (3) The backup of a gateway is selected from its neighbours having high power. (4) Non-critical backup should be preferred. (5) If non-critical backup is not available among the neighbours then critical backup is selected. (6) The backup of the gateway is the one with highest degree among its neighbors. (7) The backup of the gateway should be nearly positioned as compared to other neighbors. (8) A gateway is connected if and only if it has some neighbors. (9) The gateway nodes use long range communication interface such as Wi-Fi.

The connected nodes in the network have neighbor nodes specified as a composite object *Neighbor*. It is defined by eight fields, namely, *neighbor_id*, *type*, *backup*, *connectivity*, *pwr*, *criticality*, *neighbors* and *position*. The description of its fields is described in Table 7.

Type=<<ACTOR>>|<<SENSOR>>|<<GATEWAY>>;

Neighbor::*neighbor_id*:*Node*
type:*Type*
backup:*Neighbor*
connectivity:*Connectivity*
pwr:*Pwr*
criticality:*Criticality*
neighbors: **set of** *Node*
position:*Position*;

Table 7 Description of field of neighbour

#	Field name	Field description
1	*Neighbor_id*	Every neighbor node is unique
2	*Type*	The neighbor node may be sensor, actor or gateway
3	*Backup*	The backup is selected from neighbor nodes
4	*Connectivity*	It is required for checking the connectivity status of a node
5	*Pwr*	Power is required for backup assigning procedure
6	*Criticality*	Criticality is required for backup assigning procedure
7	*Neighbors*	Total number of neighbours is required for backup assigning procedure
8	*Position*	The position is required for backup assigning procedure as it shows the distance

Dynamic model

The dynamic model includes definition of state, functions and operations. The state of WSAN is specified as *WSAN* and comprised of five components, namely, *network_topology*, *edges*, *sensors*, *actors* and *gateways* which are described above.

```
state WSAN of
    network_topology:[NetworkTopology]
    edges:Edges
    sensors:set of Sensor
    actors: set of Actor
   gateways: set of Gateway
inv mk_WSAN(network_topology,-,sensors,actors,gateways)== card network_topology.subnets>=1 and
   forall s in set network_topology.subnets &(exists e in set s.edges & (exists s1,s2 in set sensors
   & (mk_(s1.sensor_id, s2.sensor_id)=e) and exists a1, a2 in set actors &
   (mk_(a1.actor_id, a2.actor_id)=e) and exists g1,g2 in set gateways &
   (mk_(g1.gateway_id, g2.gateway_id)=e) and (mk_(s1.sensor_id, a2.actor_id)=e) and
   (mk_(s1.sensor_id, g2.gateway_id)=e) and (mk_(a1.actor_id, g2.gateway_id)=e) and
   (mk_(a1.actor_id, a1.backup.neighbor_id) <>e) and a1.actor_id=s1.sensor_id)) and
   forall s in set network_topology.subnets &(exists nd in set s.nodes & (IsPath([nd],s)=true and
   exists ss in set sensors & exists aa in set actors & exists gg in set gateways &
   (nd=ss.sensor_id or nd=aa.actor_id or nd=gg.gateway_id)))
init w==w=mk_WSAN(nil,{},{},{},{})
end
```

Invariants (1) The network topology must employ at least one subnet. (2) For every subnet in the network topology, there exist edges which describe communication namely, sensor–sensor, actor–actor, gateway–gateway, sensor-actor, sensor-gateway and actor-gateway. (3) The neighbour of the backup of actor is not directly connected with the actor. (4) An actor can serve as a sensor as well. (5) The network is connected if there exists a path between any two nodes. The nodes may be sensors, actors or gateways. (6) All the attributes are initialized in the init function.

In a subnet, a backup for a critical actor and a gateway is required; therefore it is specified as a reusable function *AssignBackup*. This function takes critical actor as an input and returns backup as an output.

functions

AssignBackup(*critical*:*Actor*)*backup*:*Actor*

pre true

post exists *nr* **in set** *critical.actor_fields.neighbors* &

(*nr.pwr*=<<HIGH>> **and** *nr.criticality*=<<NONCRITICAL>> **and**

forall *nr1* **in set** *critical.actor_fields.neighbors* & (*nr1*<>*nr* **and card** *nr.neighbors* >=

card *nr1.neighbors*)) **and** *backup.actor_id*=*nr.neighbor_id* **or**

(**exists** *nr1* **in set** *critical.actor_fields.neighbors* & (**forall** *nr2* **in set** *critical.actor_fields.neighbors*

& (*nr1*<>*nr2* **and** (**abs** (*nr1.position.xc*-*critical.position.xc*) + **abs** (*nr1.position.yc*-*critical.position.yc*))

<=(**abs**(*nr2.position.xc*-*critical.position.xc*) + **abs** (*nr2.position.yc*-*critical.position.yc*))

and *backup.actor_id*=*nr1.neighbor_id*))) **or** (**exists** *nr* **in set** *critical.actor_fields.neighbors*

& *nr.pwr*=<<HIGH>> **and** *nr.criticality*<><<NONCRITICAL>> => *nr.criticality* =<<CRITICAL>>

and forall *nr1* **in set** *critical.actor_fields.neighbors* & (*nr1*<>*nr* **and card** *nr.neighbors* >=

card *nr1.neighbors*)) **and** *backup.actor_id*=*nr.neighbor_id* **or**

(**exists** *nr1* **in set** *critical.actor_fields.neighbors* & (**forall** *nr2* **in set** *critical.actor_fields.neighbors*

& (*nr1*<>*nr2* **and** (**abs** (*nr1.position.xc*-*critical.position.xc*) + **abs** (*nr1.position.yc*-*critical.position.yc*))

<=(**abs** (*nr2.position.xc*-*critical.position.xc*) + **abs** (*nr2.position.yc*-*critical.position.yc*))

and *backup.actor_id*=*nr1.neighbor_id*)));

Pre and post conditions: (1) There exists a node in neighbours of the critical actor that has high power, preferably non-critical and has high degree. This neighbour node is selected as a backup and it must be an actor. (2) If more than one candidate appears on the basis of power, neighbor actor status and degree then the nearly positioned actor is

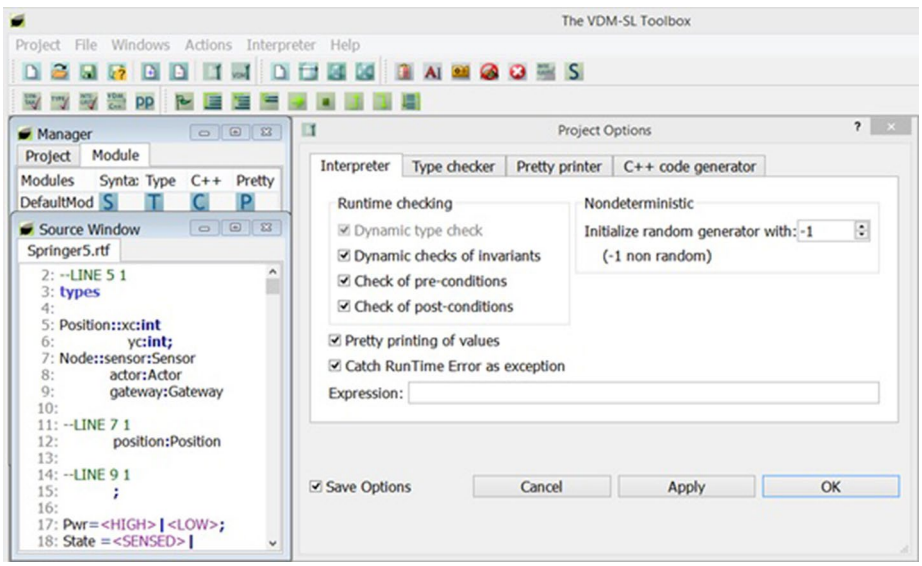

Fig. 5 Proof of correctness and dynamic checking

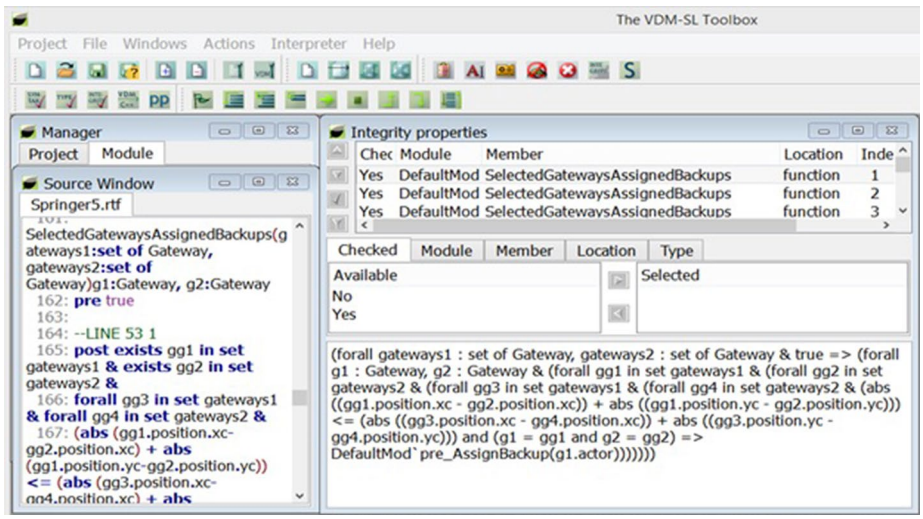

Fig. 6 Checking integrity properties

selected as a backup node. (3) If a neighbor node has high power but non-critical node is not available among the neighbours then critical node is selected to be assigned as a backup. (4) The neighbor critical node must have high degree and if more than one candidate appears on this basis then nearly positioned neighbor node is selected as a backup node.

The subnets are connected for communication through gateway nodes. A gateway node is selected from a subnet on the basis of power, degree and position which is represented as an implicit function *SamePowerDegree*. It takes two subnets as input and returns two gateways as output.

SamePowerDegree(*sub1*:*Subnet*, *sub2*:*Subnet*)*gateways1*:**set of** *Gateway*, *gateways2*:**set of** *Gateway*

pre true

post *gateways1*={*g.gateway*| *g* **in set** *sub1.nodes* & (*g.actor*. *actor_fields.pwr*=<<HIGH>> **and**

forall *g1* **in set** *sub1.nodes* & (**card** *g.actor*. *actor_fields.neighbors* >=

card *g1.actor*. *actor_fields.neighbors*))} **and** *gateways2*={*g.gateway*| *g* **in set** *sub2.nodes* &

(*g.actor*. *actor_fields.pwr*=<<HIGH>> **and forall** *g1* **in set** *sub2.nodes* &

(**card** *g.actor*. *actor_fields.neighbors* >= **card** *g1.actor*. *actor_fields.neighbors*))};

Pre and post conditions: In a subnet, there might be more than one actor nodes having high power and high degree. In other subnets, it may be possible that more than one actor nodes have high power and high degree.

Our objective is to select a gateway node from a subnet and its continuous communication with gateways of other subnets is required. That is why a selected gateway node on the basis of least position is assigned a backup within the subnet. It is expressed as an implicit function *LeastPositionedSelectedGatewaysAssignedBackups*. It takes output of the above function as input and selects a gateway from gateways on the basis of position in the subnet and then backup is assigned.

```
LeastPositionedSelectedGatewaysAssignedBackups(gateways1:set of Gateway,
        gateways2:set of Gateway)g1:Gateway, g2:Gateway
pre true
post exists gg1 in set gateways1 & exists gg2 in set gateways2 & forall gg3 in set gateways1 &
    forall gg4 in set gateways2 & (abs (gg1.position.xc-gg2.position.xc) +
    abs (gg1.position.yc-gg2.position.yc)) <= (abs (gg3.position.xc-gg4.position.xc) +
    abs (gg3.position.yc-gg4.position.yc)) and (g1=gg1 and g2=gg2) and AssignBackup(g1.actor)=gg1.actor
   and AssignBackup(g2.actor)=gg2.actor;
```

Pre and post conditions: (1) There exists a gateway node g1 in a gateway set and a gateway node g2 in another gateway set such that the absolute of the positions of g1 and g2 is less as compared to all other gateway nodes. (2) A selected gateway node from each set or subnet is also assigned backup by reusing AssignBackup function.

As the nodes in a subnet must be connected so there must exists a communication path between them. The path is specified as an explicit function *IsPath* which takes sequence of nodes and a subnet as input and outputs a Boolean value to express whether a path exists between any two nodes.

```
IsPath:seq of Node * Subnet -> bool
IsPath(nodes, sub) ==
forall n in set elems nodes & n in set sub.nodes and forall i in set inds nodes &
 i < len nodes => mk_(nodes(i), nodes(i+1)) in set sub.edges;
```

After the formal specification of functions, operations are formally specified. In our proposed algorithm, the WSAN topology is assumed to be partitioned into subnets. In a subnet, identification of critical actors is necessary because removal of a critical actor from a subnet causes the subnet to be divided into disjoint segments. That is why identified critical actors are assigned backups that will serve when critical actors fail. Its formal specification is illutrated as an operation, *CriticalsIdentificationandBackupAssigning* which outputs set of critical actors which are assigned as backups. In the operation, *network_topology* and *actors* are being read in the external clause.

```
operations
CriticalsIdentificationandBackupAssigning()criticals: set of Actor
ext rd network_topology:[NetworkTopology]
rd actors: set of Actor
pre true
post criticals={a|a in set actors & (exists s in set network_topology.subnets &
    (exists sub1,sub2 in set s.nodes & ({sub1}subset s.nodes \ {a.actor_id } and
    {sub2}subset s.nodes \ {a.actor_id } and {sub1} inter {sub2}={} and forall node1,node2 in set s.nodes
    & (node1 in set {sub1} and node2 in set {sub2} and exists nd in set s.nodes & IsPath ([nd],s) =false
    and AssignBackup(a)=a))))};
```

Pre and post conditions: The removal of a critical actor from a subnet divides a subnet into disjoint segments such that the intersection of the segments is empty. The communication path is lost between some of the nodes of a subnet and hence identified critical actors are assigned backups.

For a continuous operation of the network, there is a need to identify failure actors and to recover the operation by replacing it with the backup actors. Formal specification of *ActorFailureRecovery* is described which takes an actor as an input and evaluates failure to be true or false as output. In this operation, *network_topology* is being read and *actors* and *edges* are being written.

```
ActorFailureRecovery(actor:Actor)failure:bool
ext wr actors: set of Actor
    wr edges:Edges
    rd network_topology:[NetworkTopology]
pre forall s in set network_topology.subnets & (exists nd in set s.nodes &
    (nd.actor =actor  and actor.actor_type=<<CRITICAL>>))
post failure <=>actor.actor_fields.connectivity=<<DISCONNECTED>>
     and actor.actor_fields.neighbors={}and
     actors=actors~ \ {actor} and forall s in set network_topology.subnets &
     (exists nd in set s.nodes &  IsPath([nd],s)=false and
   let eg1= {e|e in set edges & mk_(actor.actor_id, nd)=e}
       in edges=edges~ \ eg1 and
     (actor.backup.criticality=<<NONCRITICAL>>
     => actor.backup.neighbor_id.position.xc= actor.position.xc and
      actor.backup.neighbor_id.position.yc= actor.position.yc and
      AssignBackup(actor.backup.neighbor_id.actor)=actor)
   or (actor.backup.criticality=<<CRITICAL>> and exists b in set actor.backup.neighbors &
     (b.actor=actor => AssignBackup(b.actor)=actor and actor.backup.position.xc= actor.position.xc and
     actor.backup.position.yc= actor.position.yc and b.position.xc= actor.backup.position.xc and
     b.position.yc= actor.backup.position.yc) or (actor.backup.criticality=<<CRITICAL>>
     => actor.backup.position.xc= actor.position.xc and
     actor.backup.position.yc= actor.position.yc)) and
   let eg2= {e|e in set edges & mk_(actor.backup.neighbor_id, nd)=e}
        in let eg3=eg1 union eg2
           in edges=edges~ union eg3 and IsPath([nd],s)=true
      or failure<=> actor.backup.connectivity=<<DISCONNECTED>> and actor.backup.neighbors={}
      or actors=actors~ \ {actor.backup.neighbor_id.actor} and IsPath([nd],s)=false and
   let eg4= {e|e in set edges & mk_( actor.backup.neighbor_id, nd)=e}
        in edges=edges~ \ eg4 and (actor.backup.backup.criticality=<<NONCRITICAL>>
        => actor.backup.backup.position.xc= actor.backup.position.xc and
        actor.backup.backup.position.yc= actor.backup.position.yc and
        AssignBackup(actor.backup.backup.neighbor_id.actor)=actor)
      or (actor.backup.backup.criticality=<<CRITICAL>> and exists b in set actor.backup.backup.neighbors
      & (b.actor=actor.backup.neighbor_id.actor
     and AssignBackup(b.actor)=actor and actor.backup.backup.position.xc= actor.backup.position.xc and
     actor.backup.backup.position.yc= actor.backup.position.yc
     and b.position.xc= actor.backup.backup.position.xc and b.position.yc=actor.backup.backup.position.yc)
     or (actor.backup.backup.criticality=<<CRITICAL>>
      => actor.backup.backup.position.xc= actor.backup.position.xc and
      actor.backup.backup.position.yc= actor.backup.position.yc)) and
   let eg5= {e|e in set edges & mk_(actor.backup.backup.neighbor_id, nd)=e}
      in let eg6=eg4 union eg5
         in edges=edges~ union eg6 and IsPath([nd],s)=true;
```

Pre and post conditions: (1) It is verified that the input actor is critical and exist in the subnet of the network topology. (2) In post condition, firstly the failure is detected then it is recovered. Failure occurs if and only if the connectivity of an actor is disconnected and it has no neighbour. In such a case, it will be removed from the network topology. The communication path and communication edges are lost. (3) To recover from the failure of a backup of the critical actor, non-critical is moved to the position of the critical actor. Here it becomes critical, that is why, it is assigned backup that must be an actor. (4) If failed critical actor and its critical backup both serve as backup for each other then the backup of the critical actor is assigned a new backup. The backup of the critical actor is moved to the position of the critical actor and the newly assigned backup is moved to the position of its primary actor. (5) If backup of a failed critical actor is critical then it is simply moved to the position of the failed critical actor and the backup of critical actor backup is moved to the position of the backup of the failed critical actor. (6) The communication is recovered by the union of failed actor communication edges with backup communication edges and the communication edges of the network topology are updated. (7) Similarly, failure may also occur if the backup of the critical actor fails. (8) The backup of the critical actor fails if it is disconnected and does not has neighbors then it will be removed from the actors of the network topology. (9) The communication path and communication edges are lost. (10) For the failure recovery, if the backup of the failed critical actor backup is non-critical then it is moved to the location of the failed critical actor backup. Here, it becomes critical, that is why, it is assigned backup. (11) If failed critical actor backup and its backup both serve as backup for each other then the backup of failed critical actor backup selects another actor to be assigned as a backup and moves to the position of failed critical actor backup and the newly selected backup actor is moved to the location of failed critical actor backup. (12) If backup of a failed critical actor backup is critical then it is simply moved to the position of the failed critical actor backup and the backup of failed critical actor backup is moved to the position of the backup of the failed critical actor backup. (13) The communication is recovered by the union of communication edges of failed critical actor backup with communication edges of backup of failed critical actor backup. Consequently, communication edges of the topology are updated.

Model analysis

As we know that there does not exist any computer tool which guarantees about the complete correctness of the computer model. The art to write formal specification does not provide any assurance that a model is completely correct. But if the formal specification is analysed through rigorous computer tools then it helps to identify potential errors at early stages of software development which increases a confidence of a developer.

The SFRA proposed for WSANs is formalized using VDM-SL which is a formal specification language used to analyse models both at abstract and detailed level. VDM-SL helped to examine and implement the complex model. The VDM-SL-based model helped in providing better understanding and stabilizing the requirements. Validation and verification are two main principles in the development of a system. Validation concerns with confirming whether the produced system actually fulfils user requirements

while verification ensures that the generated system in particular phase fulfils the requirements founded in the previous stage. Its purpose is to identify errors and understanding the misunderstandings at the early stages of software development and then removing these producing defects free system.

Both the static and dynamic models are analysed for the verification by syntax and type checkers that reported no errors in the specification. The syntax checker checks the syntax of the specification with respect to the definition of the VDM-SL language. The type checker identifies misuses of values and operators which can show run-time errors. The formal specification is analyzed by pretty printer which reported no error in the specification. By using facility of the Pretty Printer, the whole specification was evaluated and checked for any inconsistency in the specification. Figure 5 shows the correctness of the approach.

Although some errors may remain in the specification which are not detected by syntax/type checker and pretty printer that is why dynamic checking was enabled for checking run time errors which is shown in Fig. 5. Invariants, pre/post and dynamic checking are done to check run time errors.

The specification is analysed by integrity examiner as shown in Fig. 6. Integrity examiner checks the dynamic part of the specification and generates its series of integrity properties which if evaluate to be true then there is no run time error. It is observed that all integrity properties evaluated true of the specification. Several contradictory examples were developed for proving that our formal definition has captured the behaviour that was required. Testing and animation of the model is done through interpreter and debugger for validation which increased the confidence that the formal specification reflects the informal requirements. Several scenarios were developed for this purpose. Sensors, actors and gateways were defined and the state space of WSAN was described. Then the formal definitions analysis was done through VDM-SL Toolbox using systematic testing.

Related work

Literature review of failure recovery

Failure recovery has been studied by researchers in different contexts in WSAN. A survey of topology management techniques in wireless sensor networks to tolerate node failure is described in (Younis et al. 2014). A model of fault-tolerance is presented in (Ozaki et al. 2006), which designates multiple actors to each sensor and multiple sensors to each actor to guarantee event notification. Failure of an actor is detected and is replaced in cascaded manner (Abbasi et al. 2007). Its disadvantage is that it does not consider the criticality of a node. Topology control algorithms are proposed for providing fault-tolerance. A fault-tolerant topology is constructed by adjusting transmission power of a node. In another work, for homogeneous mobile networks, two distributed heuristics are proposed in (Ramanathan and Rosales-Hain 2000). The topology remained connected by controlling the output power of radios. A proactive distributed actor recovery algorithm (DARA) is proposed in (Abbasi et al. 2009) for distributed and large networks and is used to restore the connectivity affected due to an actor failure. To guarantee convergence, it requires more information of network state. This approach requires pre-failure planning for network partitioning. In another proactive algorithm PADRA (Akkaya

et al. 2010) connected dominating sets (CDS) are identified of the whole network. This method is not accurate for critical node identification because depth-first-search (DFS) is performed on each CDS member for the confirmation of a node that it is really a cut vertex or not. Though this algorithm is distributed but it increases message overhead because it requires 2-hop neighbours information. A reactive algorithm namely least-disruptive topology repair (LeDiR) (Abbasi et al. 2010) focuses on sustaining intra-block connectivity. In the partitioned network, it identifies smallest block and for recovering connectivity it locates the closest node to the location of failed actor in that block. In this way, the process continues in a cascaded manner to sustain intra-block connectivity. Though RIM (Younis et al. 2010), VCR (Imran et al. 2010) and C^3R (Tamboli and Younis 2010) are purely reactive and use 1-hop neighbour information to restore connectivity but could not differentiate between critical and non-critical nodes. A brief performance analysis of reactive schemes is reported in (Haider et al. 2013). It is noted that proactive approaches are impractical and reactive approaches are not suitable for complex, mission critical and safety–critical applications. That is why hybrid approaches are proposed for complex, mission critical and safety–critical applications in which critical actor proactively selects an appropriate actor to handle failure. Distributed connectivity restoration (DCR) algorithm (Akkaya et al. 2010) is hybrid which solely depends on 1-hop neighbour information increasing communication overhead. It proactively identifies critical nodes and designates appropriate backups. Similarly, a hybrid algorithm, Application-centric Recovery (ACR) is presented in (Imran et al. 2011), which identifies primary critical actors using localized information and selecting a suitable backup. The backup is selected carefully for satisfying application level of requirement.

A lot of open research challenges are introduced in WSAN for communication and coordination between sensors and actors (Cayirci 2013). An algorithm proposed in (Martirosyan and Boukerche 2012) focuses on preserving temporal relationship of events. However, it only considers actors motion and less attention is given to the sensors motion. For sensor-actor coordination a real-time coordination and routing framework is proposed in (Shah et al. 2006) for achieving reliable and energy efficient communication. In this framework, sensors form hierarchical clusters to save energy in which cluster heads communicate with actors. A survey of clustering techniques is provided in (Younis et al. 2006) for WSNs.

Literature review of CAS using formal methods

Though some work has been done on modeling of multi-agents systems using formal methods but still further investigation is needed to use formal methods in combination with complex networks and multi-agent methodologies to model the complex adaptive systems. The framework based on unified modeling language and architecture description language of multi-agent systems is presented in (Park and Sugumaran 2005) which used service agent communication that recognizes knowledge query manipulation language. Open Agent Architecture is proposed in (Martin et al. 1999), which used Inter-agent Communication Language using agent as a facilitator. To establish a link between mathematical models and biological systems an approach based on process algebra is used for agents based communication (Sumpter and Blanchard 2001). Wireless sensor networks are complex networks (Batool et al. 2014; Kumar et al. 2015) and have

applications ranging from military to environmental implementations. A formal model of wireless sensor networks in combination with agent-based simulation model is presented in (Niazi and Hussain 2011a, b). As this work lacks mathematical modelling that is why it is extended in (Chaudhry 2015) using Gaussian function for sensing of emergent behaviour in CAS. Experiments have been carried out to deduce centrality metrics effects to validate the roles of nodes in complex networks (Batool and Niazi 2014). Both formal model and visual-agent based complex network representations are made for the cognitive evolution in the form of a temporal cognitive level networks (Hussain and Niazi 2014). The effectiveness of the approach is validated using historic data of citations. Agent-based & complex network-based methods are used for modelling complex adaptive systems (Niazi and Hussain 2011c, 2012; Niazi 2013). To guarantee safety, Petri-nets are employed to model railway interlocking components (Khan et al. 2011, 2014; Khan and Zafar 2011; Zafar 2011). Z-notation and X-machine are used for the formal specification of multi-agent systems with a dynamic behaviour and structures (Ali et al. 2012). VDM-SL and Z based formal models are developed for WSANs for safety critical systems (Imran et al. 2015; Alnuem et al. 2014; Riaz et al. 2015).

Conclusions

Complex adaptive systems (CAS) are dynamic networks having many agents which act and react constantly in response to each other. The CAS control is decentralized which is required to be coherent and accomplished by many decisions made by a large number of agents in competition with each other. Mostly CAS are tested by computer based simulation techniques using agent-based methodologies and complex networks. Simulation techniques cannot verify a complete correctness of a system because the number of test cases increases exponentially to gain a required level of confidence. Formal methods help to overcome the disadvantages of simulation and increase a confidence over the developed models (Afzaal and Zafar 2015). Therefore, formal methods in terms of VDM-SL are used in this work to specify and prove correctness of the proposed model.

WSANs are complex adaptive which are modelled as dynamic undirected graphs. Sensors, actors and gateways are represented as nodes and communication links between the nodes are represented as edges in graph based model. In dynamic graphs, the topology changes frequently because the nodes are mobile and become connected or disconnected frequently. As the graph is undirected, that is, if a node n1 can communicate with node n2 then the node n2 can also communicate with the node n1. That is why defining existence of path between any two nodes of the network was easier and economical in terms of time complexity. WSANs are described using graph based models because these are effective for storing and processing information of any kind of networks.

Our subnet-based failure recovery algorithm (SFRA) is energy efficient because the recovery process is localized. That is SFRA is efficient because of the advantage of partitioning of WSANs into subnets localizing the failure recovery procedure at subnet level which reduces computation of the algorithm. The partitioning of WSAN into subnets conserves energy as the cascading process becomes limited at subnet level. SFRA is hybrid as it assumes pre-failure planning and post failure recovery which is suitable for mission critical and security critical applications. In pre-failure planning, it selects a gateway node in a subnet required for connecting and communication among subnets

and identifies critical nodes in a subnet. Then the gateway nodes and critical nodes are assigned backups. In post-failure recovery, backup replaces the primary failed node and invokes the failure recovery of the backup node recursively.

Most of the work on modeling WSANs is simulation based which have various disadvantages. To overcome these limitations, formal techniques are required which assure about correctness of the models. That is why SFRA is formalized using Vienna development method-specification language (VDM-SL) which is used at abstract level. Further, VDM-SL has a detailed descriptive power for validation and verification of the specification. In this work, static part of the specification was presented using several data types and the dynamic model was described as a state space, functions and operations. Invariants were defined on data types to define the criteria for a safe behaviour of objects and pre/post conditions were used to insure safety so that the system should not enter into any unwanted situation. The proposed model is analysed using the existing facilities in the VDM-SL Toolbox.

We know the importance of testing, simulations and experimentation for performance evaluation of the network which will be considered in our future work for the further visualization of the results.

Abbreviations

WSANs: wireless sensor and actor networks; VDM-SL: Vienna development method-specification language; SBFR: subnet-based failure recovery algorithm; CAS: complex adaptive systems; ABM: agent-based methodologies; CN: complex networks; DARA: distributed actor recovery algorithm; CDS: connected dominating sets; DFS: depth-first-search; LeDiR: least-disruptive topology repair; DCR: distributed connectivity restoration; ACR: application centric recovery.

Authors' contributions

HA and NAZ have proposed the localized, hybrid and energy-efficient Subnet-based Failure Recovery Algorithm (SFRA) for Wireless Sensor and Actor Networks (WSANs). The authors have proposed the novel approach of subnets in WSAN which is more energy efficient as compared to the clustering technique. Graph theory is used to model WSANs topology to use the graph based structures for efficient storage and for processing the network by graph based algorithms. To overcome the disadvantages of simulations authors have transformed the algorithm into an equivalent formal specification using Vienna Development Method Specification Language (VDM-SL) to prove its correctness. They have analyzed, validated and verified the developed formal specification through VDM-SL Toolbox. Both authors read and approved the final manuscript.

Authors' information

Ms. Hamra Afzaal is Postgraduate student at Computer Science Department, COMSATS Sahiwal, Pakistan. Her research interests are formal methods, wireless sensor and actor networks, integration of approaches, etc.

Nazir A. Zafar was born in 1969 in Pakistan. He received his M.Sc. (Math. in 1991), M. Phil (Math. in 1993), and M.Sc. (Nucl. Engg. in 1994) from Quaid-i-Azam University, Pakistan. He was awarded PhD degree in computer science from Kyushu University, Japan in 20 04. He has served at various universities and well-reputed scientific organizations in Pakistan. For example, he has worked (2010–2014) as Associate Professor at the College of Computer Sciences and Information Technology (CCSIT), King Faisal University (KFU), Al Ahsa, Saudi Arabia. He has also worked (2007–2010) as Dean/Professor of Faculty of Information Technology, University of Central Punjab, Lahore, Pakistan. Currently, he is working as a Professor & Head at the Department of Computer Science, COMSATS Sahiwal, Pakistan. His research interest includes modelling of systems using formal approaches, integration of approaches, safety critical systems. He is an active member of Pakistan Mathematical Society. He has contributed for scientific and technical committees including organizing conferences and curriculum development in the capacity of a member as well as chairman.

Competing interests

Both authors declare that they have no competing interests.

Received: 10 January 2016 Accepted: 19 October 2016
Published online: 18 November 2016

References

Abbasi AA, Akkaya K, Younis M (2007) 32nd IEEE conference on local computer networks (LCN), a distributed connectivity restoration algorithm in wireless sensor and actor networks, pp 496–503

Abbasi AA, Younis M, Akkaya K (2009) Movement-assisted connectivity restoration in wireless sensor and actor networks. IEEE Trans Parallel Distrib Syst 20(9):1366–1379

Abbasi AA, Younis M, Baroudi U (2010) Restoring connectivity in wireless sensor-actor networks with minimal topology changes. IEEE international conference on communications (ICC), pp 1–5

Afzaal H, Zafar NA (2015) 6th international conference on information and communication technologies (ICICT), formal modeling and algorithm of subnet-based backup assigning in WSAN

Afzaal H, Zafar NA (2015) 9th international conference on open source systems and technologies (ICOSST), formal localized reactive subnet-based failure recovery model for sparsely connected wireless sensor and actor networks

Afzaal H, Zafar NA (2015) 1st international conference on dependable embedded wireless and sensing networks (DEWSNet), algorithm and formal specification of subnet-based communication in WSANs

Afzaal H, Zafar NA (2016) 1st national conference on trends and innovations in information technology (TIIT), centralized confidentiality-based formal algorithm in WSANs

Afzaal H, Imran M, Zafar NA (2015) 13th international conference on frontiers of information technology (FIT), implementing partitioning detection and connectivity restoration in WSAN using VDM-SL

Akkaya K, Senel F, Thimmapuram A, Uludag S (2010) Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. IEEE Trans Comput 59(2):258–271

Ali G, Khan S, Zafar NA, Ahmad F (2012) Formal modeling towards a dynamic organization of multi-agent systems using communicating X-machine and Z-notation. Indian J Sci Technol 5(7):2972–2977

Alnuem M, Zafar NA, Imran M, Ullah S, Fayed M (2014) Formal specification and validation of a localized algorithm for segregation of critical/non-critical nodes in MAHSNs. Int J Distrib Sens Netw 1–14. doi:10.1155/2014/140973

Armano G, Javarone MA (2013) Clustering datasets by complex networks analysis. Complex Adapt Syst Model 1(5):1

Batool K, Niazi MA (2014) Towards a methodology for validation of centrality measures in complex networks. Plos ONE 9(4):e90283

Batool K, Niazi M, Sadik S, Shakil ARR (2014) Towards modeling complex wireless sensor networks using agents and networks: a systematic approach, IEEE conference on TENCON, pp 1–6

Borozan V, Ferrara M, Fujita S, Furuya M, Manoussakis Y, Stolee D (2016) Partitioning a graph into highly connected subgraphs. J Graph Theory 82:322–333

Bouarfa S, Blom HA, Curran R, Everdij MH (2013) Agent-based modeling and simulation of emergent behavior in air transportation. Complex Adapt Syst Model 1(1):1–26

Boulaire F, Utting M, Drogemuller R (2015) Dynamic agent composition for large-scale agent-based models. Complex Adapt Syst Model 3(1):1

Cayirci E (2013) Wireless sensor and actuator network applications and challenges. Auton Sensor Netw 1–15

Chaudhry QA (2015) A Gaussian function model for simulation of complex environmental sensing. Complex Adapt Syst Model 3(1):1–4

Fortino G, Trunfio P (2014) Internet of things based on smart objects, technology, middleware and applications. Springer, Berlin

Fortino G, Guerrieri A, O'Hare GMP, Antonio GR (2012) A flexible building management framework based on wireless sensor and actuator networks. J Netw Comput Appl 35(6):1934–1952

Fortino G, Galzarano S, Gravina R, Li W (2015) A framework for collaborative computing and multi-sensor data fusion in body sensor networks. Inform Fusion 22:50–70

Fu X, Li W, Fortino G (2013) Empowering the invulnerability of wireless sensor networks through super wires and super nodes, CCGRID

Haider N, Imran M, Saad NM, Zakariya M (2013) Performance analysis of reactive connectivity restoration algorithms for wireless sensor and actor networks, IEEE Malaysia international conference on communications (MICC)

Holland John H (2006) Studying complex adaptive systems. J Syst Sci Complex 19(1):1–8

Hussain A, Niazi M (2014) Toward a formal, visual framework of emergent cognitive development of scholars. Cogn Comput 6(1):113–124

Imran M, Zafar NA (2012) Formal specification and validation of a hybrid connectivity restoration algorithm for wireless sensor and actor networks. Sensors 12(9):11754–11781

Imran M, Younis M, Said AM, Hasbullah H (2010) Volunteer-instigated connectivity restoration algorithm for wireless sensor and actor networks, IEEE international conference on wireless communications, networking and information security (WCNIS), pp 679–683

Imran M, Said AM, Younis M, Hasbullah H (2011) Application-centric connectivity restoration algorithm for wireless sensor and actor networks, advances in grid and pervasive computing. Springer, Berlin, pp 243–253

Imran M, Zafar NA, Alnuem MA, Aksoy MS, Vasilakos AV (2015) Formal verification and validation of a movement control actor relocation algorithm for safety–critical application. Wireless Netw 22:247–265

Khan SA, Zafar NA (2011) Improving moving block railway system using fuzzy multi-agent specification language. Int J Innov Comput Inform Control 7(7B):4517–4533

Khan SA, Zafar NA, Ahmad F (2011) Petri net modeling of railway crossing system using fuzzy brakes. Int J Phys Sci 6(14):3389–3397

Khan SA, Zafar NA, Ahmad F, Islam S (2014) Extending Petri net to reduce control strategies of railway interlocking system. Appl Math Model 38(2):413–424

Kumar PR, Wainwright MJ, Zecchina R (2015) Mathematical foundations of complex networked information systems. Politecnico di Torino, Verrès, Italy, 2141, Springer

Martin DL, Cheyer AJ, Moran DB (1999) The open agent architecture: a framework for building distributed software systems. Appl Artif Intell 13(1–2):91–128

Martirosyan A, Boukerche A (2012) Preserving temporal relationships of events for wireless sensor actor networks. IEEE Trans Comput 61(8):1203–1216

Niazi MA (2013) Complex adaptive systems modeling: a multidisciplinary roadmap. Complex Adapt Syst Model 1(1):1–14
Niazi M, Hussain A (2011a) Sensing emergence in complex systems. IEEE Sens J 11(10):2479–2480
Niazi M, Hussain A (2011b) A novel agent-based simulation framework for sensing in complex adaptive environments. IEEE Sens J 11(2):404–412
Niazi M, Hussain A (2011c) Agent-based computing from multi-agent systems to agent-based models: a visual survey. Scientometrics 89(2):479–499
Niazi MA, Hussain A (2012) Cognitive agent-based computing-I: a unified framework for modeling complex adaptive systems using agent-based & complex network-based methods. Springer, Berlin
North MJ (2014) A theoretical formalism for analyzing agent-based models. Complex Adapt Syst Model 2(3):1
Ozaki K, Watanabe K, Itaya S, Hayashibara N, Enokido T, Takizawa M (2006) IEEE 20th international conference on advanced information networking and applications (AINA), a fault-tolerant model for wireless sensor-actor system, p 5
Park S, Sugumaran V (2005) Designing multi-agent systems: a framework and application. Expert Syst Appl 28(2):259–271
Ramanathan R, Rosales-Hain R (2000) Topology control of multihop wireless networks using transmit power adjustment. In: Proceedings of nineteenth annual joint conference of the IEEE computer and communications societies (INFOCOM), vol. 2, pp 404–413
Riaz S, Afzaal H, Imran M, Zafar NA, Aksoy MS (2015) Formalizing mobile ad hoc and sensor networks using VDM-SL. Procedia Comput Sci 63:148–153
SCSK Corporation (2013) VDM Tools, User Manual, Version 9.0.2
SCSK Corporation (2013) VDM Tools, Language Manual, Version 9.0.2
Shah GA, Bozyiğit M, Akan ÖB, Baykal B (2006) Real-time coordination and routing in wireless sensor and actor networks. In: Next generation teletraffic and wired/wireless advanced networking. Springer, Berlin, pp 365–383
Shah MA, Abbas G, Dogar AB, Halim Z (2015) Scaling hierarchical clustering and energy aware routing for sensor networks. Complex Adapt Syst Model 3(1):1–23
Sumpter DJT, Blanchard GB (2001) Ants and agents: a process algebra approach to modelling ant colony behavior. Math Biol 63(5):951–980
Tamboli N, Younis M (2010) Coverage-aware connectivity restoration in mobile sensor networks. J Netw Comput Appl 33(4):363–374
Younis O, Krunz M, Ramasubramanian S (2006) Node clustering in wireless sensor networks: recent developments and deployment challenges. IEEE Netw 20(3):20–25
Younis MF, Lee S, Abbasi AA (2010) A localized algorithm for restoring inter-node connectivity in networks of moveable sensors. IEEE Trans Comput 59(12):1669–1682
Younis M, Senturk IF, Akkaya K, Lee S, Senel F (2014) Topology management techniques for tolerating node failures in wireless sensor networks: a survey. Comput Netw 58:254–283
Zafar NA (2011) Formal dynamic operational model of RIS components. IJCSNS 11(9):91–96
Zafar NA (2016) Formal specification and analysis of take-off procedure using VDM-SL. Complex adaptive systems modelling