Rigorous Design of Fault Tolerance and Recovery Algorithm for Disaster Management and Relief Distribution System using Event-B

Pooja Yadav1*, Raghuraj Suryavanshi2 and Divakar Yadav3
1Dr A P J Abdul Kalam Technical University, Lucknow 226 031, India
2Pranveer Singh Institute of Technology, Kanpur 209 305, India
3Institute of Engineering and Technology Lucknow 226 021, India

Received 25 January 2022; revised 25 April 2022; accepted 25 April 2022

India is vulnerable to disasters such as earthquakes, floods, tsunamis, landslides forest fires and cyclones due to its unique socio-economic and geo-climatic conditions. Twenty seven out of thirty-six states and union territories are prone to different types of disasters which cause loss of life, disruption of livelihoods, damage to infrastructure and property which in turn becomes a heavy burden on the national economy. Effective management of relief work is a key step towards normalizing human life post disaster. In this paper, we have presented the formal development and verification of a fault tolerance and recovery algorithm for district level disaster control centers in India which are connected to each other via a communication network. Formal methods help in the verification of critical properties of complex systems by developing mathematical models so that design errors can be detected and removed during the early stages of software development. Event-B, which is a formal method and Rodin platform is used for this work. Event-B is a mathematical language of first-order logic to provide a solution to the complex algorithms formally. In this algorithm a Disaster Control Centre is chosen as the coordinator based on its unique vote value. This vote value is allotted and modified dynamically based on the extent of damage in the area where the center is located. The center having the highest vote value among the currently active centers is elected as the coordinator. The correctness of the algorithm is verified through discharge of proof obligations generated by the Event-B model.

Keywords: Event-B, Formal methods, Fault tolerance, Proof obligations, Verification

Introduction

In India, the probability of occurrence of disasters is compounded due to increasing environmental degradation, unplanned urbanization, geological hazards, increased development in high-risk zones, deforestation, climate change etc. 58.6% of the country’s landmass is vulnerable to moderate to high intensity earthquakes, 12% is prone to flooding, 5700 km out of 7516 km of coastline is vulnerable to tsunamis and cyclones, 15% of the landmass which includes hilly areas is prone to landslides as per the Annual Report 2020-21 of the National Disaster Management Authority.1 These disasters pose a serious threat to the country’s population, economy, and sustainable development. These frequently occurring natural disasters cause loss of lives, property, livestock, crops etc. The government has shifted from a relief centric approach to disaster management towards a more holistic approach involving prevention using early warning, mitigation, relief, and rehabilitation. The logistic support during relief work involves deployment of boats, aircrafts, Central Armed Police Forces (CAPFs), special units of Armed Forces, National Disaster Response Force (NDRF), essential commodities and relief materials and restoration of critical infrastructure facilities including communication network as required to manage the situation effectively. The number of casualties and economic losses can be significantly reduced if relief reaches the affected population within a stipulated period during the time of disaster. This includes both evacuation work and supply of essential amenities such as food, clean drinking water, medicines etc. During such times there are a large number of distress calls using mobile phones, social media and any other means of communication available to the people stranded in remote locations. The government sets up helplines and relief centers to manage these distress calls and to set up a supply chain for evacuation work and to provide relief material such as clean drinking water, food, medicines, cloths etc.

In this paper, we have tackled the above problem by developing an algorithm for fault tolerance and recovery for district level disaster control centers. In
this algorithm, a Disaster Control Center is declared as a coordinator. Although the working of all the Disaster Control Centers is similar, only one center is chosen as the coordinator to coordinate the activities of all the centers. It has the task of allotting work to other disaster control centers, routing distress calls to the center which is nearest to the affected area, managing resources and supply chain for relief materials. This is done to ensure that all the affected areas get equitable relief and resources. If all disaster control centers work independently than relief material and resources may reach some affected areas multiple times while some affected areas may not receive any relief material and resources. The coordinator prevents such a situation from arising. The coordinator may fail to respond due to a variety of reasons such as communication network failure, fire, building collapse or any other untoward situation during the time of a disaster. Other Disaster Control Centers may become directionless due to failure of coordinator and relief work may be severely hampered. The proposed fault tolerance algorithm is used for the selection of a new coordinator in case of failure of the current coordinator and recovery of the failed coordinator. The notion of vote allotment is used for the purpose of selection of the new coordinator. A vote value is allotted to each a Disaster Control Center or node which is based on the severity of damage at its geographical location. The node where the intensity of the disaster is least is allotted the highest vote value because this node has the least chance of failure. This vote value is increased or decreased dynamically as the latest information regarding the intensity of the disaster at each geographical location is updated. As soon as a Disaster Control Center or node detects the failure of the coordinator, it sends a request message to all the active nodes requesting their vote value. Upon receiving the vote value from all active nodes, the requesting node declares the node with the highest vote value as the coordinator. The algorithm has been rigorously specified through Event-B using the eclipse-based Rodin platform. The correctness of the algorithm is verified through generation and discharge of proof obligations.

**Related Work**

Several tools, algorithms and mobile based applications have been developed the purpose of providing automation in the field of disaster management which have helped in expediting the process of disaster mitigation and relief work. Gollet et al. give an overview of the challenges encountered while managing the supply chain for relief materials after the occurrence of a natural disaster. A mathematical model for distribution of relief materials in optimum time using multi-objective optimization algorithms is developed by Sadeghi et al. The use of social media for coordinating relief work in Indian cites has been demonstrated by Basu et al. The paper identifies the need and availability of resources from social media and uses pattern matching techniques for mapping the requirement and availability of rescue and logistic equipment. Fajardo & Oppus demonstrate the development of an android based application for determining optimum routes for supplying relief materials to affected areas. Route optimization has been obtained by the application of genetic algorithms on the Travelling Salesman Problem. However, the application of formal methods in the field of disaster management and relief work is relatively unexplored. In this paper, a formal model for fault tolerance for Disaster Control Centers in India has been developed and verified using Event-B which provides an exhaustive approach for the specification of formal models for algorithms based on distributed systems using mathematical constructs. The formal verification of lazy replication in distributed database systems is demonstrated by Suryavanshi & Yadav. The formal verification of distributed checkpointing is highlighted by Chandra et al. The checkpointing algorithm can be used for recovery from failure. Recently formal techniques have been used for the development and verification of practical application-based systems because formal methods help in the verification of critical property of complex systems by developing mathematical models so that design errors can be detected and removed during the early stages of software development. Butler & Yadav highlight the formal development of an incremental model of Mondex system using Event-B. Mondex is a smart card based electronic cash system which was developed by Graham Higgins and Tim Jones of the National Westminster Bank in the United Kingdom. The Event-B model of receiver-initiated load balancing algorithm in distributed systems is outlined by Yadav et al. In this algorithm, the lightly loaded site initiates the process of load balancing by trying to obtain load from a heavily loaded site. Bourbouh et al. demonstrates the
applicability of formal methods for the verification of autonomous robotic systems using the inspection rover case study. Morris et al.\textsuperscript{11} presents a refinement based Event-B model for the formal verification of “run to completion” state chart models which are used for designing complex controllers which react to environmental triggers by running a sequential process. Our paper demonstrates the formal development and application of leader election algorithm\textsuperscript{12,13} in distributed systems for fault tolerance in integrated Disaster Control Centers in India.

Methodology

Event-B: A Formal Modelling Approach

Event-B\textsuperscript{14,15} supports a refinement-based framework for formal modelling of distributed system algorithms. An Event-B model consists of two types of components: contexts and machines.\textsuperscript{16} Contexts consist of sets, axioms and constants which form the static part of the model. Sets can be carrier or enumerated. The properties of constants and sets are described by axioms. The dynamic or behavioral part of the model is represented by machines which comprise of variables, theorems, invariants, and events.\textsuperscript{17} A machine may see the context either directly or indirectly and its state is defined by the variables. The variables are bound by certain conditions or constraints which are defined by invariants. These invariants must not be violated during execution when the state of the machine changes. Every state of the machine must satisfy all the invariants. The violation of invariants indicates that the model is not working as per the specifications and needs to be corrected.\textsuperscript{18} The context and invariants of the machine are used for the derivation of theorems. A machine also consists of several events which describe the evolution of the state of the machine. Guards and actions together constitute an event. Guards are the necessary conditions which must be satisfied for an event to be enabled and the action listed in the event to be performed.\textsuperscript{19} These actions modify the value of variables through deterministic or non-deterministic substitution. The initial state of the model is defined by the initialization event which does not have any guard. The correctness and properties of the model are verified by discharge of proof obligations.\textsuperscript{20} The proof obligations may be discharged either manually or automatically.

The Event-B notations are described in depth by Abrial.\textsuperscript{15} However, some of the notations of Event-B used frequently in the proposed model for fault tolerance algorithm are given in Table 1 and explained below:

- Let $X$ and $Y$ be two sets, the operator $\leftrightarrow$ specifies the relation between $X$ and $Y$ such that, $X \leftrightarrow Y = P(X \times Y)$ where $\times$ is the cartesian product of $X$ and $Y$.
- If a relation $R \in X \leftrightarrow Y$ exists, then the mapping of elements $x$, $y$ such that, $x \in X$ and $y \in Y$ will be given as $x \mapsto y$.
- If a relation $R \in X \rightarrow Y$ exists, then the domain of relation $R$ is the set of elements of $X$ that are related to some of the elements of set $Y$ by relation $R$. It is defined as, $\text{dom}(R) = \{x|x \in X \land \exists y \in Y | f(x, y)\}$
- If a relation $R \in X \leftarrow Y$ exists, then the range of relation $R$ is the set of elements of $Y$ that are related to some element of set $X$. It is specified as, $\text{ran}(R) = \{y|y \in Y \land \exists x \in X | f(x, y)\}$
- A relation with some special properties is known as a function. A partial function from set $X$ to set $Y$ ($X \rightarrow Y$) is defined as a relation of an element of set $X$ with at most a single element in set $Y$. A partial function $f \in X \rightarrow Y$ is specified as, $\forall (x, y_1, y_2), \{x \in X \land y_1 \in Y \land y_2 \in Y \Rightarrow (x \mapsto y_1 \in f \land x \mapsto y_2 \in f) \Rightarrow y_1 = y_2\}$
- A total function from set $X$ to set $Y$ written as $f \in X \rightarrow Y$, is a partial function such that $\text{dom}(f) = X$. Given that $f \in X \mapsto Y$ and $x \in \text{dom}(f)$, $x$ is mapped to a unique value by $f$ which is represented by $f(x)$.

This work has been carried out using the Rodin platform which is an eclipse based tool for rigorous specification and verification of Event-B models.

<table>
<thead>
<tr>
<th>Table 1 — Some Event-B notations frequently used in the proposed Event-B model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
System Model

In this paper, earthquake has been used as an example for demonstrating the process of vote allotment and selection of a new coordinator upon the failure of the existing coordinator. The process will remain similar for other natural or man-made disasters. The proposed model for integrated Disaster Control Centres is shown in Fig. 1. In this algorithm, it is assumed that every Disaster Control Centre or node has a unique vote value. The term node and DCC has been used interchangeably while describing the algorithm. The DCC which has the highest vote value is chosen as the coordinator. For choosing the node with the highest vote value, any node (suppose DCC_k) which knows that the coordinator has failed, sends a broadcast message to all the participating or active nodes. After receiving the vote value from all the active DCCs, the requesting DCC finds the highest vote value. The DCC with the highest vote value is declared as the coordinator and a message is broadcast to inform every other node about the new coordinator. The vote value of each node is decided on the basis of its proximity to the epicentre of the earthquake and will be updated dynamically as and when the latest information about the intensity of the earthquake and aftershocks at the geographical location of the node is available. An informal description of events of the algorithm as per Table 2 is as follows:

**Allotment of Vote Value**

Epicenter of an earthquake is point where the intensity of the earthquake is maximum. Initially a

<table>
<thead>
<tr>
<th>Table 2 — Algorithm for Coordinator Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all DCC_1 to DCC_n</td>
</tr>
<tr>
<td>DCC_k detects failure of coordinator</td>
</tr>
<tr>
<td><strong>Processing at DCC_k</strong></td>
</tr>
<tr>
<td>Broadcast vote_request message to all DCCs</td>
</tr>
<tr>
<td><strong>Processing at other DCCs</strong></td>
</tr>
<tr>
<td>Receive vote_request message</td>
</tr>
<tr>
<td>Send vote_reply message with vote value V_i</td>
</tr>
<tr>
<td><strong>Processing at DCC_k</strong></td>
</tr>
<tr>
<td>Receive vote_reply message with vote value V_i</td>
</tr>
<tr>
<td>Find V_max</td>
</tr>
<tr>
<td>Extract DCC having V_max (Sender of the vote_reply message with vote value V_max)</td>
</tr>
<tr>
<td>Declare DCC having V_max as new coordinator</td>
</tr>
<tr>
<td><strong>Processing at Coordinator</strong></td>
</tr>
<tr>
<td>Broadcast i_am_coordinator message to all DCCs</td>
</tr>
<tr>
<td><strong>Processing at other DCCs</strong></td>
</tr>
<tr>
<td>Receive i_am_coordinator message</td>
</tr>
<tr>
<td>Process any task allocated by the coordinator</td>
</tr>
</tbody>
</table>

END

**Fig. 1 — Proposed model for integrated disaster control centers**
node or disaster control center which farthest from the epicenter is allotted the highest vote value as this node is assumed to be least prone to damages such as building collapse, fire, network, and power failures etc. The distance of a node from the epicenter is calculated as the Euclidean distance\(^5\) between two coordinates \((p_1, q_1)\) and \((p_2, q_2)\) is
\[
d = \sqrt{(p_2 - p_1)^2 + (q_2 - q_1)^2}
\]
where, \((p_1, q_1)\) are the coordinates of the epicenter and \((p_2, q_2)\) are the coordinates of the node. Initially, the vote value allotted to all the nodes is based on the Euclidean distance between their geographical location and the epicenter of the earthquake. The node with the highest Euclidean distance from the epicenter is allotted the highest vote value and vice versa.

**Upgradation of Vote Value**

The process of vote increment and decrement at each node is done based on the seismic map representing the intensity of the earthquake in various affected areas. Intensity of an earthquake is measured on the Richter scale on a magnitude of 1 to 10. An example of a seismic intensity map is shown in Fig. 2.

The directions of major axis and minor axis of the seismic map is derived from the distribution of fault zones and aftershocks.\(^{21}\) The derivation of length of minor axis and major axis in each intensity circle is demonstrated by Sun et al.\(^{22}\) This seismic map is updated as and when new information is received from the affected areas. The length of the axis of an intensity circle is changed when the intensity at an area as per the latest information is different from the previously recorded intensity which also affects the vote value of a node which is inversely proportional to the recorded intensity. The boundary of the intensity circle is expanded or reduced at the area where new intensity value is recorded. Suppose there is an area \(x\) which currently lies in the \((R-2)\) intensity circle, but as per the latest information the intensity at area \(x\) is \((R-1)\) on Richter scale. In this case, the boundary of \((R-1)\) intensity circle must be expanded. The length of the minor and major axis will be changed according to
\[
L'_{R-1} = L_{R-1} + \mu(L_{R-1} - L^*)
\]
where, \(L_{R-1}\) is the new length of axis of intensity circle \((R-1)\), \(L'_{R-1}\) is the current length of axis of intensity circle \((R-1)\), \(L^*\) is the length of the axis of the intensity circle in which area \(x\) currently lies and \(\mu\) is the learning rate which is calculated using the least mean square method for training historical data.\(^{23}\)

This seismic intensity map is used for the purpose of vote allotment. If a node or Disaster Control Center lies in the outer intensity circle than it is allotted a higher vote value as compared to a node which lies in the inner intensity circle. This ensures that the node which is most prone to damage has the least chance of becoming the coordinator. If the intensity circle of a node changes as per Eq. (2) the vote value will be increased or decreased accordingly.
\[
v_1 \alpha (L'_{R-1} - L_{R-1})
\]
where, \(v_1\) is the adjustment factor for vote value,
\(L'_{R-1}\) is the new length of axis of intensity circle \((R-1)\), \(L_{R-1}\) is the old length of axis of intensity circle \((R-1)\).

**Failure of the Coordinator**

If the coordinator DCC fails due to any reason viz. communication network failure, building collapse, fire, flooding etc. the process for selection of a new coordinator is started.

**Broadcasting of Request Message**

A request message for obtaining vote values of all active nodes is broadcast by a node on learning that the existing coordinator has failed, and a new coordinator must be selected.

**Receiving of Request Message**

This request message is received by all other nodes which are in an active state.

**Reply to Request Message with Vote Value**

Every DCC has a vote value which is a natural number and is assigned as per the criteria described in Eqs (1) – (3). All the DCCs send a reply message with their vote value corresponding to the request message.
Receiving of Request Message with Vote Value
The requesting DCC receives these reply messages from all other DCCs containing their vote value.

Choosing the Coordinator
After receiving the vote values of all the active nodes, the requesting node chooses the highest vote value among them. The node with the highest vote value is declared as the coordinator.

Broadcasting of Coordinator Declaration Message to All the DCCs
After being selected as the coordinator, the node must broadcast the message “I_am_coordinator” to all the participating nodes.

Receiving of the Coordinator Declaration Message
The message “I_am_coordinator” will be received by all the participating nodes. Now all the participating DCCs are informed about the newly elected coordinator.

Recovery of Failed Coordinator/DCC
The procedure for recovery is same for all failed DCCs whether they are coordinator or not. A failed DCC is allotted the vote value of zero after it recovers from failure. This is done to prevent a failed DCC from becoming the coordinator immediately after recovery. The vote value of the recovered DCC is later updated according to Eq. (2) as and when new information regarding intensity of earthquake is obtained.

Formal Description of Events
The fault tolerance algorithm is formalized using Event-B with the help of the eclipse-based Rodin platform. The machine is the dynamic part of the model while the context is the static part of the model. Machine contains invariants and events. The machine part sees the context part which is contains constants, sets and axioms. In the context section, DCC and MESSAGE are carrier sets and message_type and dcc_type are enumerated sets. In the axiom section, the message_type is defined in three ways, vote_request, vote_reply or i_am_coordinator. The values for the enumerated set dcc_type can be active, failed or pending. The context part of the model is not changed until the change of the formal requirements. The invariants of the model are defined below:

inv1: sender ∈ MESSAGE → DCC
inv2: mess_type ∈ MESSAGE → message_type
inv3: deliver ∈ DCC → MESSAGE
inv4: message_vote_val ∈ MESSAGE → N

inv5: vote_msg_send ∈ (MESSAGE ↔ MESSAGE) → DCC
inv6: vote_msg_rcv ∈ DCC → (MESSAGE ↔ MESSAGE)
inv7: vote_msg ∈ MESSAGE → N
inv8: vote_count ∈ [N]
inv9: dcc_state ∈ DCC → dcc_type
inv10: coordinator ∈ DCC
inv11: vote ∈ DCC → N
inv12: counter ∈ N
inv13: causaldelivery ∈ DCC ↔ MESSAGE
inv14: causaldelivery ∈ MESSAGE ↔ MESSAGE
inv15: ordereddelivery ∈ DCC ↔ (MESSAGE ↔ MESSAGE)
inv16: sentmessages ∈ MESSAGE

Initialization
In the machine part, all the variables are initialized with 0 or ∅. Counter is initialized with 0 because it must be a natural number N. Other variables are initialized with ∅ (Empty value). Initialization of dcc_state for each DCC is “active”.

Invariants
In invariant 1, the variable sender defines the broadcast of vote_request message to all the DCCs in the set DCC as the mapping function {mm↦ pp} ∈ sender which denotes that the vote_request message mm has been successfully sent by the DCC pp. In invariant 2, the variable mess type defines the type of message. The message can be vote_request, vote_reply or i_am_coordinator. In invariant 3, the variable deliver shows the delivery of the request message to all the DCCs except the sender of that message. In invariant 4, the variable message_vote_value defines the vote value of every DCC which is assigned at the time of initialization of the DCC. In invariant 5, the variable vote_msg_send ensures that all participating DCCs except the receiving DCC and failure DCCs send the vote_reply message. In invariant 6, the variable vote_msg_rcv shows the delivery of the vote_reply message from all the participating DCCs which is denoted by the function (pp↦ {mm↦ mr}) ∈ vote_msg_rcv. This mapping function means that the DCC pp receives the vote_reply message mr corresponding to the vote_request message mm. In invariant 7, the variable vote_msg defines the message with the vote value sent by the participating DCCs. In invariant 8, the variable vote_count is defined as a set of natural numbers comprising of the vote values of participating DCCs received with the vote_reply message. Invariant 9 defines the variable dcc_state as the state of a DCC. State of a DCC can be active, pending or failed. The variable coordinator is a subset of the set DCC as per invariant 10. The DCC with the highest vote_count is...
chosen as the coordinator. In invariant 11, the variable vote defines the vote value of each DCC at the time of vote allotment or vote decrement which is a natural number. The value of the variable counter must be a natural number as per invariant 12. The variable counter is used to ensure that the requesting DCC has received the vote values from all the participating DCCs. The value of counter is increased by one each time a vote value is received by the requesting DCC.

The variable causaldelivery represents causal order based delivery of messages at any DCC as per invariant 13. In invariant 14, the variable causalorder represents the causal precedence among the messages. Message msg1 causally precedes the message msg2 is presented by the mapping \((msg1 \mapsto msg2) \in causalorder\). In invariant 15, the variable ordereddelivery shows the delivery order of messages at any DCC. Message msg1 is delivered before message msg2 at DCC \(dd\) is shown by the mapping \(dd \mapsto (msg1 \mapsto msg2) \in ordereddelivery\). Invariant 16 specifies the variable sentmessages as the set of messages which have already been sent.

Event 1: Allotment of Vote Value

When a fresh DCC is started, the vote value of that DCC is assigned with a natural number. The vote value of a node or DCC is inversely proportional to its proximity to the epicenter of the earthquake which is calculated as per Eq. (1). This ensures that the node which is most vulnerable to damage has the least chance of becoming the coordinator.

EVENT Vote Allotment
ANY \(dd, vt\)

WHERE 
\(grd1: dd \in DCC\)
\(grd2: vt \in \mathbb{N}\)

THEN 
act1: vote(dd) := vt

END

The event of vote allotment is shown above. The variable \(dd\) belongs to the set \(DCC\) and \(vt\) is a natural number is ensured by the guards \((grd1&2)\). In the action part, act 1 assigns the vote value \(vt\) to the DCC \(dd\).

Event 2: Modification of Vote Value

This event is triggered when any DCC decreases the vote partially from its vote value. A similar procedure is followed when the vote value of a DCC needs to be incremented. The vote value at a node is modified dynamically based on the intensity of disaster at the location of the node as per Eq (3). The vote value is decreased if it is known that the intensity of disaster at the geographical location of the node is greater than the previously recorded intensity and vice versa. The event for vote decrement is shown below.

EVENT Vote Decrement
ANY \(pp, vt\)

WHERE
\(grd1: dd \in DCC\)
\(grd2: vt \in \mathbb{N}\)
\(grd3: vt < vote(dd)\)

THEN 
act1: vote(dd) := vote(dd) − vt

END

DCC \(dd\) belongs to the set \(DCC(grd1)\). The value \(vt\) is the adjustment factor for vote value and it is a natural number \((grd2)\). Guard 3 ensures that \(vt\) is less than vote value of \(dd\) so that the vote value of \(dd\) does not become less than zero after decrement. In the action part, the vote of the DCC \(dd\) is decreased by \(vt1(\text{act1})\).

Event 3: Failure of the Coordinator

The event models the failure of the coordinator. The failure of the coordinator can disrupt the relief work being carried out by other Disaster Control Centres or nodes. This is because the coordinator allot work to all nodes which is required for the efficient management of distress calls and disaster relief operations. A new coordinator must be chosen immediately so the relief work continues without any interruption.

EVENT Failure of Coordinator
ANY \(d\)

WHERE
\(grd1: [d]=\text{coordinator}\)
\(grd2: dcc\_state(d)=\text{active}\)

THEN 
act1: dcc\_state(d) := \text{failed}
act2: coordinator := coordinator \(\setminus d\)

END

DCC \(d\) is the coordinator and state of DCC \(d\) is active \((grd1&2)\). When the DCC \(d\) fails, the dcc\_state of \(d\) is set as “failed”(act1). After failure, DCC \(d\) is removed from the set coordinator(act2).

Event 4: Broadcast of Request Message

For selecting a coordinator, any DCC can broadcast the request message for receiving the vote value from every other DCC.
EVENT Broadcast Request Message
ANY dd, mm

WHERE
grd1: dd∈ DCC
grd2: mm∈ MESSAGE
grd3: mm∈ dom(sender)
grd4: mm∈ sentmessages
grd5: coordinator = \emptyset
grd6: dcc_state(dd)=active

THEN
act1: sender := sender∪{mm↦ dd}
act2: mess_type(mm) := vote_request
act3: sentmessages := sentmessages∪{mm}
act4: causalorder := causalorder∪{sender·{(dd)× {mm}}∪{delivered{(dd)×{mm}}})
act5: causaldelivery := causaldelivery∪{(dd)×{mm})
act6: ordereddelivery := ordereddelivery∪{(dd↦ causaldelivery{(dd)×{mm})}

END

The event for broadcast of request message is demonstrated above. Guards 1 and 2 represent DCC dd and message mm belong to the sets DCC and MESSAGE respectively. Message mm has not been sent (grd3&4). The set coordinator is empty (grd5) and the state of DCC dd is “active” (grd6). If all the guards become true, the DCC dd will broadcast the message mm and type of the message is vote_request (act1&2). Action 3 adds message mm to the set of sent messages. This event also ensures that the vote request message is delivered at the sending site according to causal order. All the messages broadcast by DCC dd causally precede message mm (act4). The delivery of message mm at DCC dd is specified by action 5 and the delivery order of message mm is given by action 6.

Event 5: Receiving of Request Message

The request message is broadcast by the sender DCC to all the participating DCCs. This event given below models the receiving of request message by a participating DCC.

EVENT Receive Request Message
ANY mm, d

WHERE
grd1: d∈ DCC
grd2: mm∈ dom(mess_type)
grd3: mess_type(mm) = vote_request
grd4: {d↦ mm})∈ causaldelivery
grd5: dcc_state(d)=active
grd6: ∀ mm∈ MESSAGE A (mm↦ mm)∈ causalorder⇒ (d↦ m)∈ causaldelivery

THEN
act1: causaldelivery := causaldelivery∪{d↦{mm))
act2: ordereddelivery := ordereddelivery∪{d↦{deliver{(dd)×{mm})}}

END

DCC d is in the set DCC(grd1). Message mm belongs to the domain of the sender (grd2). Type of message mm is vote_request and message mm is not yet delivered to the participating DCC d(grds3&4). The state of DCC dd is “active” (grd5). Guard 6 ensures that all the messages msg which causally precede message mm have already been delivered at DCC d. If all the guards are enabled, then message mm will be successfully received at DCC d(act1). The delivery order of message mm at DCC d is specified by action 2.

Event 6: Sending of reply message

After receiving the request message, all the participating DCCs send the vote_reply message with their vote value.

EVENT Send Reply Message
ANY dd, mr, mm, d, vn

WHERE
grd1: dd∈ DCC
grd2: d∈ DCC
grd3: mm∈ dom(mess_type)
grd4: mess_type(mm) = vote_request
grd5: (mm↦ dd)∈ sender
grd6: (d↦ mm)∈ deliver
grd7: mr∈ MESSAGE
grd8: [mr↦ mm]∈ dom(vote_msg_send)
grd9: vn=vote(d)
grd10: (mr↦ vn)∈ vote_msg
grd11: dcc_state(d)=active
grd12: dcc_state(dd)=active
grd13: ∀ mm∈ MESSAGE, d∈ DCC A (mm↦ mm)∈ causalorder

THEN
act1: mess_type(mm) := vote_reply
act2: vote_msg_send := vote_msg_send∪{(mr↦ mm)⇒ d]
act3: message_send(mr) := vote(d)
act4: vote_msg := vote_msg∪{mr⇒ vn}
act5: sentmessages := sentmessages∪{mr}

END

The event given above demonstrates the generation of reply of vote_request message by participating nodes. Request message mm belongs to the domain of mess_type and type of the message is vote_request(grds3&4). Message mm is broadcast by the DCC dd is ensured by guard 5. Message mm is successfully delivered to DCC d(grd6). Message mr is
in the set MESSAGE (grd7). Reply message mr, which corresponds to the vote request message mm is not yet sent by the participating DCC d(grd8). The vote value of the DCC d is vn, which is a natural number and vote_reply message mr with vote value vn is not yet sent (grd9 &10). The state of DCCs d and dd is “active” (grd11&12). The vote request message mm causally precedes all other vote request messages msg (grd13). DCC d will send vote reply message corresponding to the vote request message mm only. This prevents the coordinator selection algorithm from being run on multiple DCCs and prevents network congestion due to excess reply messages. In the action part, the message type of mr is vote_reply(act1). Vote_reply message mr, which corresponds to the vote_request message mm is sent by DCC d (act2). Vote of DCC d is assigned to message_vote_value of message mr(act3). Vote_reply message mr with vote value vn is in the set vote_msg(msg)(act4). Message mr is added to the set of sent messages (act5).

**Event 7: Delivery of Reply Message**

Vote_reply message is sent by all the participating DCCs to the requesting DCC. The requesting DCC receives the vote values in the vote_reply message.

WHERE

<table>
<thead>
<tr>
<th>grd1: dd ∈ DCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>grd2: d ∈ DCC</td>
</tr>
<tr>
<td>grd3: mm ∈ MESSAGE</td>
</tr>
<tr>
<td>grd4: mr ∈ MESSAGE</td>
</tr>
<tr>
<td>grd5: mess_type(mm) = vote_request</td>
</tr>
<tr>
<td>grd6: mess_type(mr) = vote_reply</td>
</tr>
<tr>
<td>grd7: (mm ↦ dd) ∈ sender</td>
</tr>
<tr>
<td>grd8: (d ↦ mm) ∈ deliver</td>
</tr>
<tr>
<td>grd9: (mm ↦ mr) ↦ d ∈ vote_msg_send</td>
</tr>
<tr>
<td>grd10: dd ↦ (mm ↦ mr) ∈ vote_msg_rcv</td>
</tr>
<tr>
<td>grd11: d_cc_state(dd) = active</td>
</tr>
<tr>
<td>grd12: d_cc_state(d) = active</td>
</tr>
</tbody>
</table>

THEN

| act1: vote_msg_rcv := vote_msg_rcv U {dd ↦ (mm ↦ mr)} |
| act2: vote_count := vote_count U {message_vote_val(mr)} |
| act3: counter := counter + 1 |

END

In the event modelled above, the DCCs d and dd belong to the set DCC(grd1 &2). DCC dd is the requesting DCC and DCC d is a participating DCC. The message type of messages mm and mr is vote_request and vote_reply respectively (grd5 &6). Vote_request message mm is broadcast by DCC dd(grd7). Delivery of vote_request message mm to the participating DCC d is successful is ensured by guard 8. Vote_reply message mr with the vote value is sent successfully from the participating DCC dd(grd9). Vote_reply message mr is not yet received by the requesting DCC dd(grd10). State of the DCC d and dd is “active” is ensured by guards 11 & 12.

If all the guards are true, then the vote_reply message mr, which corresponds to the vote_request message mm is delivered at DCC dd(act1 & 2). The vote value of the participating DCC d which was received by the requesting DCC dd with the vote_reply message mr is added to the set vote_count (act2). The value of the variable counter is increased by 1 (act3). The variable counter ensures that the vote value of all the participating DCCs has reached the requesting DCC.

**Event 8: Selection of the Coordinator**

After successfully receiving the vote values from all the participating DCCs, the DCC with the maximum vote value is chosen as the coordinator among the participating DCCs. As the vote value of a node is inversely proportional to the intensity of disaster at its location, the chosen coordinator will be least prone to damage and the relief work can proceed effectively without any disruption.

WHERE

<table>
<thead>
<tr>
<th>grd1: dd ∈ DCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>grd2: d ∈ DCC</td>
</tr>
<tr>
<td>grd3: mm ∈ dom(mess_type)</td>
</tr>
<tr>
<td>grd4: mess_type(mm) = vote_request</td>
</tr>
<tr>
<td>grd5: mr ∈ dom(mess_type)</td>
</tr>
<tr>
<td>grd6: mess_type(mr) = vote_reply</td>
</tr>
<tr>
<td>grd7: (dd ↦ (mm ↦ mr)) ∈ vote_msg_rcv</td>
</tr>
<tr>
<td>grd8: max_vote_count ∈ ℕ</td>
</tr>
<tr>
<td>grd9: max_vote_count = max (vote_count U vote(pp))</td>
</tr>
<tr>
<td>grd10: (mr ↦ max_vote_count) ∈ vote_msg</td>
</tr>
<tr>
<td>grd11: mr ∈ dom(sender)</td>
</tr>
<tr>
<td>grd12: sender(mr) = d</td>
</tr>
<tr>
<td>grd13: counter = card (DCC) – 1</td>
</tr>
<tr>
<td>grd14: d_cc_state(dd) = active</td>
</tr>
<tr>
<td>grd15: d_cc_state(d) = active</td>
</tr>
</tbody>
</table>

THEN

| act1: coordinator := {d} |

END

In the above event, the DCCs dd and d are in the set DCC(grd1 &2). The messages mm and mr belong to the domain of mess_type (grd3&5). The type of message mm and mr is vote_request and vote_reply.
respectively (grd4 &6). Vote_reply message mr, which corresponds to the vote_request message mms delivered at DCC dd(grd7). The value of the variable max_vote_count must be a natural number and is chosen as the maximum value from the set vote_count(grd8 &9). Now the vote_reply message mr whose vote value is equal to the max_vote_count is chosen from the set vote_msg(grd10). Guard 11 ensures that vote_reply message mr is in the domain of the sender. Guard 12 checks that the sender of vote_reply message mr (whose vote value is equal to the max_vote_count) is DCC d. Guard 13 ensures that the requesting DCC has received the vote value of all the participating DCCs. State of the DCC d and dd is “active”(grd14&15). If all the guards are true, then the DCC d is declared as the coordinator (act1).

Event 9: Broadcasting of the Coordinator Selection Message to All the participating DCCs

After selecting the DCC with the highest vote value, the coordinator selection message is broadcast to all the participating DCCs so that all the DCCs have knowledge about the coordinator.

EVENTBroadcast the coordinator selection message to all DCCs

WHERE

ANYd, mm

WHERE

grd1:{d}=coordinator

grd2:mm ∈ MESSAGE

grd3:mm ∈ dom(sender)

THEN

act1:sender= senderU {mm→ d}

act2:mess_type(mm)= i_am_coordinator

END

In the event modelled above, DCC d is the coordinator, message mm is in the set MESSAGE and message mm is not in the domain of sender(grd1, 2 &3). In the action part, the DCC d broadcasts the message mm and its category is “i_am_coordinator” (act1 &2).

Event 10: Receiving of the coordinator selection message

The coordinator selection message is received by all the participating DCCs. Now, the new coordinator will allot work to these participating DCCs.

EVENTReceive the coordinator selection message

WHERE

ANY d, mm, dd

END

In the above event, DCC d is the coordinator and DCC dd is in the set DCC (grd1 &2). Message mm belongs to the domain of the sender and the type of message mm is i_am_coordinator(grd3 &4). Message mm is not yet delivered at the DCC dd(grd5). If all the guards are enabled, then message mm will be received successfully at all the participating DCCs dd(act1).

Event 11: Recovery of a Failed Coordinator/ Disaster Control Centre

The failed coordinator is treated as an any ordinary DCC for the purpose of recovery. However, on recovery the recovered node or DCC is allotted a vote value of zero. This ensures that the failed coordinator or DCC does not become the coordinator immediately after recovery since it might be prone to failure. The vote value of the recovered DCC is updated as and when the latest information regarding the intensity of the earthquake and aftershocks is received. The event models the recovery of a failed DCC.

EVENTRecovery of failed DCC

WHERE

ANYd

END

In the event modelled above, DCC d is in the set DCC and state of DCC d is failed (grds1 &2). In the action part, the dcc_state of d is set as “active” and the vote value of DCC d is set as zero (act1 &2).

Results and Discussion

In this paper, we have demonstrated the formal development of fault tolerance and recovery procedure for disaster management and response system. This work is conducted using the Rodin platform which provides a framework for the development of Event-B models. The correctness of the model is verified through generation and discharge of proof obligations. These proof obligations help in refinement and consistency checking to ensure the safety and liveness properties of the model.
• Safety property for the proposed algorithm states that the coordinator is always in an active state. This is ensured by the following invariant.

**Invariant 13:** \( \forall d \in DCC \land \{d\} = \text{coordinator} \land d \in \text{dom}(dcc\_state) \Rightarrow dcc\_state(d) = \text{active} \)

• Liveness property for the proposed fault tolerance algorithm states that the DCC with the highest vote value is chosen as the coordinator. The vote value of all other DCCs is less than the vote value of the coordinator is ensured by the following invariant.

**Invariant 14:** \( \forall d, dc \in DCC \land \{d\} = \text{coordinator} \Rightarrow \text{vote}(d) > \text{vote}(dc) \)

The algorithm satisfies the following essential properties:

• **Uniqueness:** Exactly one DCC is declared as the coordinator at a particular point of time. The following invariant is added to ensure that only one node is the coordinator. Invariant 15 implies that the cardinality of the set coordinator will always be one.

**Invariant 15:** \( \text{card}\{\text{coordinator}\} = 1 \)

• **Agreement:** When a DCC is chosen as the coordinator it broadcasts the message “\text{i\_am\_coordinator}” to all the participating DCCs. Thus, all the DCCs have a common knowledge about the coordinator.

• **Termination:** A new coordinator is selected within a finite time using the proposed algorithm.

### Complexity Analysis

A fault tolerance and recovery algorithm for District level Disaster Control Centres is proposed in this paper. Since all the DCCs communicate with each other through message passing, the complexity is calculated using the cost of communication which is based on the number of messages required by the algorithm for selection of the coordinator. The communication cost is calculated in the following manner:

Number of participating DCCs in the system = \( n \)

When a DCC starts the process of selecting a coordinator, it broadcasts a request message to all the DCCs which requires \((n - 1)\) messages.

On receiving the vote request message, all the participating DCCs will send a reply message which requires \((n - 1)\) messages.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Total POs</th>
<th>P0s Discharged Automatically</th>
<th>P0s Discharged Manually</th>
<th>Undischarged POs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCC context</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DCC Machine</td>
<td>40</td>
<td>25</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

On selection of the coordinator, the message “\text{i\_am\_coordinator}” is broadcast to all the participating DCCs which uses \((n - 1)\) messages.

Thus, the total number of messages utilized by the algorithm will be \((n - 1) + (n - 1) + (n - 1) = 3(n - 1)\).

Therefore, the complexity of the proposed algorithm is \( O(n) \).

An Event-B model of fault tolerance for integrated district level Disaster Control Centres has been developed using the eclipse-based platform called Rodin. Rodin platform provides an exhaustive package for management of proofs for models based on distributed system algorithms. The correctness of the model has been verified through generation and discharge of proof obligations (POs) which are either discharged by automatic provers provided by Rodin or through manual intervention. These proof obligations are generated due to invariants which give a deep insight into the system properties and help us understand why a design decision is correct. The discharge of proof obligations also helps us to create new invariants leading to a more concrete system specification. The proof statistics of the model are given in Table 3.

The algorithm is specified with the help of invariants and events. The algorithm is formalized step by step using events which consist of guards and actions. Proof obligations are generated and discharged using the provers of the Rodin tool for refinement and consistency checking of the proposed model. All the proof obligations generated by the proposed model are discharged without any anomaly which establishes the correctness of the model.

### Conclusions

Natural disasters occur frequently in India and the damage caused by these disasters is aggravated by the fact that India is the second most populated country in the world and has a high population density. These natural disasters not only lead to immediate loss of life and property, but also leave a long-lasting impact on the livelihoods of the population living in disaster.
hit zones due loss of crops, cattle, fertile soil, and public and private infrastructure. We have demonstrated a fault tolerant system for integrated district level Disaster Control Centres which can be deployed in disaster prone zones for effective disaster mitigation and relief management. The novelty of the work lies in the application of formal methods in the field of disaster management. The correctness of the model is verified through discharge of proof obligations. The proposed model can be used effectively for fault tolerance at integrated district level Disaster Control Centres for uninterrupted service. In future we propose to extend the proposed fault tolerance algorithm for integration of Covid-19 Control Rooms and supply chain for oxygen, equipment, and medicines first at state level and then at national level.

References