Using Event-B Method to model Routing Information Protocol

Bahija Boulamaat, Anas Amamou, Rajaa Filali, Sanae El Mimouni and Mohamed Bouhdadi

***Abstract*—**Event-B is a formal method used for software development, in complex system. It uses the notion of refinement step by step modeling, going from an abstract level and add more details gradually to move on to a more concrete one. This method use proof obligations rules which based on mathematical approach to prove the consistency and the correctness of the modeling. This paper presents an incremental formal modeling of the Routing Information Protocol (RIP) using event-B method. Routing Information Protocol is a standard based, distance vector, interior gateway protocol used by routers to exchange routing information among gateways and other hosts. It plays an important role providing the shortest and best path for data to take from node to node. It permits to update periodically the routing information in the RIP network.

***Keywords*—** Event B, Formal modeling, RIP, refinement.

I. Introduction

The routing information protocol (RIP) is one of the internal gateway protocols (interior gateway protocol), it manages the route information in a local network, it considered as the effective solution for small networks. The RIP is a dynamic protocol used to find the best route or path from end-to-end over a network by using a routing metric/hop count algorithm. This algorithm is used to determine the shortest path from the source to destination.

The first version of the routing information protocol is defined by IETF in RFC 1058 [7] published in 1988, it is considered a classful routing protocol. Due to the many limits of the version 1, the second version developed in RFC 2453 published in 1993 [8] and standardized in 1998 [9], it is considered a classless routing protocol, and has the ability to carry subnet information. The third version of RIP called RIPng (RIP next generation) developed in RFC 2080 is an extension of the RIPv2 to support IPv6.

Bahija Boulamaat is with LMPHE laboratory, University of Mohammed V, Faculty of sciences, Rabat, Morocco (e-mail: boulamaatbahija@gmail.com).

Anas Amamou is with LMPHE laboratory, University of Mohammed V, Faculty of sciences, Rabat, Morocco (e-mail: amamou.anas@yahoo.fr).

Rajaa Filali is with LMPHE laboratory, University of Mohammed V, Faculty of sciences, Rabat, Morocco (e-mail: rajaafilali@gmail.com).

Sanae El Mimouni is with LMPHE laboratory, University of Mohammed V, Faculty of sciences, Rabat, Morocco (e-mail: sanae.elm@ gmail.com).

Mohamed Bouhdadi is with LMPHE laboratory, University of Mohammed V, Faculty of sciences, Rabat, Morocco (e-mail: bouhdadi@fsr.ac.ma).

This paper presents the informal specification of RIP which will be modeled using event-B method. This article is an extended version of a conference paper that appeared as [1].

Event B is a formal method for modeling software systems [2]. It has been used in several interdisciplinary such as medicine, transport, aeronautics, trains, space, biology, security, hybrid system, parallel systems and communications protocols…etc. It translates an informal specification to a formal notation using mathematical language (elementary set theory, first order logic...etc). It goes from developing a discrete system by refinement. This method permits to build a model by successive steps going from an abstract model to a more concrete one adding more details gradually. Each version is proved and is consistent with the previous one [3].

The refinement notion was first introduced by Dijikstra [3] in the 70s, and then formalized with Back [13], then it was mentioned in several research such as Lamport [14],Abrial,..

Jean Raymond Abrial has developed the Event-B in 2010 which is the evolution of the B method developed in the 90s and also the Z language which is developed in the seventies. It works essentially on the concepts of: refinement, composition and generecity [5]. The advantage of this method is to make proofs automatically using the Rodin platform. The Even-B use the Rodin Platform that is an Eclipse based IDE that provides refinement and mathematical proofs. The proofs help to improve the failure and errors and help up to correctly model.

The reminder of this paper is as follows. In Section2, we give an overview of the Event-B method. In Section 3, we present the description of the routing information protocol. And, the final section presents the modeling of the protocol.

II. Overview of the Event-B Method

Event-B is a formal method used to model complex systems. It uses mathematical language to built models step by step going from an abstract one to a refined one and proving it the correctness. The Event-B models consist of two mean constructs: the contexts and the machines. The context contains the static part of the model like sets and constants, and axioms whereas the machines contain the dynamic part like variables, invariants and events. Between the machines and contexts, there are different relationships. The machines can refine one or several ones. The contexts can be extended by one or several context and can be referenced ‘’see’ by one or several machines.

In Event-B, an event is defined by the syntax: EVENT e WHEN G THEN S END , Where G is the guard, expressed as a first-order logical formula in the state variables, and S is any number of generalized substitutions, defined by the syntax S ::= x := E(v) | x := z : | P(z). The deterministic substitution, x := E(v), assigns to variable x the value of expression E(v), defined over set of state variables v. In a non-deterministic substitution, x := z : | P(z), it is possible to choose non-deterministically local variables, z, that will render the predicate P(z) true. If this is the case, then the substitution,
x := z, can be applied, otherwise nothing happens.

The Event-B consists of three important techniques: refinement, composition, instantiation.

Refinement: the refinement permits to build model gradually by making it more and more accurate. We construct the models by sequence; each one is the refinement of the previous one in the sequence. The refinement uses the concept of the superstition refinement and data refinement.

Decomposition: the decomposition consists on spitting a model to small sub models.

Generic instantiation: the generic instantiation permit to parameterize machines in order to reuse it to refinement. It consists of using a generate theory proved using constants and axioms in a machines to be reused in another machine without proving it.

The proof obligation permits to test and validate the model. The proof obligation rules define what must be, it correct the error and help to ameliorate the model. They verify the properties of the machines and ensure the correctness of the modeling and its consistency between the refined and the abstract levels. The proof obligation rules define what must be proven; verify certain properties of the machines. Rodin platform generates automatically this proof with the help of the proof obligation generator. There several proof obligation as INV, FIS, WD, EQL...

Invariant preservation proof obligation rule (INV) ensure that each invariant are preserved by each event, the Feasibility proof obligation rule (FIS) ensure that the action are feasible. The well-definedness proof obligation rule (WD) ensures that a potentially ill-defined axiom, theorem, invariant, guard, action, variant, or witness is indeed well defined

III. Description of RIP

Routing Information Protocol (RIP) is a standard distance-vector protocol used by routers to exchange routing information between each entity: host, gateway, within the network. Each entity participating in the routing scheme sends update messages that describe the routing information currently exist in that entity.

The RIP is used to find the best route or path from end-to-end (source to destination) over a network by using a routing metric/hop count algorithm. This protocol is used to determine the shortest path from the source to destination. Hop count is the number of nodes the packet must go through until it reaches its destination [7], [8], [9], the routing information is stored in a routing table for future use.

The routing information protocol manages the router information by enable to exchange routing information in network (RFC 1058). RIP allows connecting with destinations that is not directly reachable. It used for a finite number of nodes the longest path supported is 15.if the metric's (the metric is the number of node from the originating node to reach a destination) node value is 16 the destination is considered unreachable. Each node in the network should know all the routes to the other nodes .the routing table is updated periodically to ensure the freshness of the routes.

A node sends a broadcast request to RIP neighbors' interfaces in the network, the request consist of its entire routing table. All the neighbors receiving the request respond with their routing tables, and these tables will also be sent to the receiving neighbors.

The node send a request to its neighbors, each node of the neighbors will send its routing tables to its neighbors until all the nodes in the network have all the routes in their tables.

The messages received, permit to update the receiving 'node table. Each entry in the routing table presents a route, it consist of source, destination and the metric node to reach the destination. The update consist of comparing the received table with the table of the node ,If an entry is in the received node routing table but it does not exist in the node routing table than we add it as a new one in the node table. If it already exists we take the minimum metric, as it considered the best path to the destination.

The routing information table has four important timers, which used in the routing information protocol as

The Route update timer: each 30 second a router send a copy of its routing table to its neighbors.

Route invalid timer: a time to determine when a route is considered invalid, when there is no update for a route about 180 s then the router sends updates to its neighbors that the route is invalid.

Route hold-down timer: is the time in which a route in unreachable, is about 180s or until a better route is found

Route flush timer: is the time to remote a route from the routing table, it comes after the route is considered invalid.

The limits specifications of this protocol are: The hop count cannot exceed 15, otherwise it will be considered unreachable, RIP has slow convergence and has count to infinity problems, Variable Length Subnet Masks are not supported by RIP version 1.

IV. Model in event-B

In the initial model, we present the exchange between a node and its neighbors. We take the node and one neighbor because the same pattern is repeated with the other nodes. The source node send messages to it neighbor, this messages consist of the source node, routing table, and then it received a response messages from the neighbor, also consist of the neighbors routing table. In first model we also present the update of the source node table as how it updated. The update happens when we compare the source node routing table and the received node routing table. If there are routes in the received table that are not in the source table, we then add those routes as new entries. If the same routes exist in both tables, we consider the route with best path (with the minimum metric).

A. *The initial model*

We consider the message exchange between the nodes. We model it as an exchange between one node and its neighbors; in the RIP network .We called the source node and one of its neighbors we called it a neighbor node

In the context:

We define two carrier sets: NODE and MSG.

In the machine:

Sd,and rcv present the messages send or received by a node in the RIP network, we also define the routing table by the variable entrytable, and hopecount who presents the routing table of the resource node (rcvnode)and respectively entrytablercv, and hopecountrcv presents the routing table of the neighbors node (neibornode),

```
CONTEXT   ctx0
SETS   NODE,MSG
END
```

```
VARIABLES
   Sd,rcv,srcnode, neibornode,
   entrytable, entrytablercv,
   hopecount, hopecountrcv
INVARIANTS
   inv1:  $sd \subseteq MSG$
   inv2:  $rcv \subseteq MSG$
   inv3:  $srcnode \in MSG \nrightarrow NODE$
   inv4:  $neibornode \in MSG \nrightarrow NODE$
   inv5:  $entrytable \in NODE \leftrightarrow NODE$
   inv6:  $entrytablercv \in NODE \leftrightarrow NODE$
   inv7:  $hopecount \in NODE \nrightarrow (NODE \nrightarrow \mathbb{N})$
   inv8:  $hopecountrcv \in NODE \nrightarrow (NODE \nrightarrow \mathbb{N})$
   inv9:  $\forall i,j \cdot i \mapsto j \in entrytable \Rightarrow i \neq j$
   inv10: $\forall i,j \cdot i \mapsto j \in entrytablercv \Rightarrow i \neq j$
```

In the event initialization we initialize the variables defined previously:

```
INITIALISATION:
   THEN
      act1: $sd := \emptyset$
      act2: $rcv := \emptyset$
      act3: $srcnode := \emptyset$
      act4: $neibornode := \emptyset$
      act5: $entrytable := \emptyset$
      act6: $hopecount := \emptyset$
      act7: $entrytablercv := \emptyset$
      act8: $hopecountrcv := \emptyset$
   END
```

We have two events for the exchange messages between a source node and one of it neighbors. And three events for the update of the routing table of the source node.

The event send_request and receiv_request present the messages send or received by a node:

```
send−request:
   ANY
      msg, s , n
   WHERE
      grd1:  $msg \in MSG$
      grd2:  $s \in NODE$
      grd3:  $n \in NODE$
      grd4:  $msg \in dom(srcnode)$
      grd5:  $msg \in dom(neibornode)$
      grd6:  $srcnode(msg)=s$
      grd7:  $neibornode(msg)=n$
      grd8:  $s \neq n$
   THEN
      act1: $sd := sd \cup \{msg\}$
   END
```

```
receiv−request:
   ANY
      msg   s   n
   WHERE
      grd1:  $msg \in MSG$
      grd2:  $s \in NODE$
      grd3:  $n \in NODE$
      grd4:  $msg \in dom(srcnode)$
      grd5:  $msg \in dom(neibornode)$
      grd6:  $srcnode(msg)=s$
      grd7:  $neibornode(msg)=n$
      grd8:  $s \neq n$
   THEN
      act1: $rcv := rcv \cup \{msg\}$
   END
```

After we compare the routes in the route table of the source node and the one received from the neighbor node. We update the source routing table, when the route in the received table does not exist in the resource routing table (event updare_table−dif), or when the route exists in the routing table but the received one has a better path because it has an inferior count metric (update_table_same_entry1). Finally, there is no update when the route in the source routing table has an inferior metric node than the one in the received one; in this case there is no action so it’s a SKIP, nothing changing in the resource routing table.

```
updare_table−dif:
   ANY
      s, n,   msg1,  msg2
   WHERE
      grd1:  $msg1 \in MSG$
      grd3:  $msg2 \in MSG$
      grd3:  $s \in NODE$
      grd4:  $n \in NODE$
      grd5:  $s \neq n$
      grd6:  $msg1 \in dom(srcnode)$
      grd7:  $msg2 \in dom(neibornode)$
      grd8:  $srcnode(msg1)=s$
      grd9:  $neibornode(msg2)=n$
      grd10: $s \mapsto n \notin entrytable$
      grd1:  $n \mapsto s \in entrytablercv$
   THEN
```

```
      act1: entrytable≔ entrytable ∪ {s↦n}
    END

  update_table_same_entry1:
    ANY
      s , n , msg1, msg2
    WHERE
      grd1:  msg1 ∈ MSG
      grd2:  msg2∈MSG
      grd3:  s∈NODE
      grd4:  n∈NODE
      grd5:  msg1∈dom(srcnode)
      grd6:  n≠s
      grd7:  msg2∈dom(neibornode)
      grd8:  srcnode(msg1)=s
      grd9:  neibornode(msg2)=n
      grd10: n↦s∈entrytablercv
      grd11: s↦n∈entrytable
      grd12: s∈dom(hopecount)
      grd12: n∈dom(hopecount(s))
      grd13: n∈dom(hopecountrcv)
      grd14: s∈dom(hopecountrcv(n))
      grd15: hopecount(s)(n)>hopecountrcv(n)(s)
THEN
      act1: entrytable ≔ entrytable ∪ {n↦s}
    END

  update_table_same_entry2:
    ANY
      n ,s , msg1, msg2
    WHERE
      grd1:  msg1 ∈ MSG
      grd2:  msg2∈MSG
      grd3:  s∈NODE
      grd4:  n∈NODE
      grd5:  msg1∈dom(srcnode)
      grd6:  n≠s
      grd7:  msg2∈dom(neibornode)
      grd8:  srcnode(msg1)=s
      grd9:  neibornode(msg2)=n
      grd10: n↦s∈entrytablercv
      grd11: s↦n∈entrytable
      grd12: s∈dom(hopecount)
      grd13: n∈dom(hopecount(s))
      grd14: n∈dom(hopecountrcv)
      grd15: s∈dom(hopecountrcv(n))
      grd16: hopecount(s)(n)<hopecountrcv(n)(s)
END
```

B. *The first refinement*

In the first refinement, we add three more events related to the four timers. Then we refine the previous events by adding more guard.

Route update (t1), time route invalid (t2), time, route flush time (t3) and time route hold-down (t4).

The initial context will be extended and the initial machine will be refined.

Firstly we extend the initial context, by defining the states of the route if it is valid or unreachable (the metric is above 15) or invalid and adding the timers.

```
CONTEXT
    ctx01
 EXTENDS
    ctx0
SETS
    ETAT
CONSTANTS
    Valid,invalid ,unreachable
    t1,t2,t3,t4
AXIOMS
    ETAT={valid,invalid,unreachable}
    valid ≠ unreachable
    unreachable≠invalid
    t1 ∈ ℕ ,t2 ∈ ℕ,t3∈ℕ ,t4∈ℕ
END
```

In this refinement we have two more variables a time: temps and a Boolean for the first timer.

```
      VARIABLES
        b,temps,etat

      INVARIANTS
          inv1:  temps ∈ ℕ
          inv2:  b ∈ BOOL
          inv3:  b= TRUE ⇒ temps <t1
          inv4:  etat ∈ ETAT
```

We add in the INITIALISATION event the initialization of the new variables

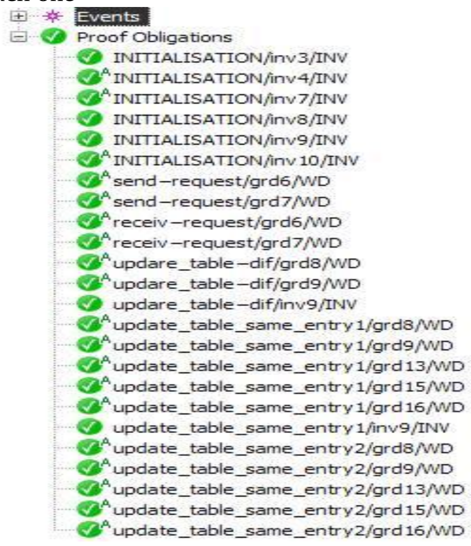
```
      act9: temps ≔ 0
      act10: b ≔ FALSE
      act10: etat ≔ valid
```

The previous events are extended in this refinement by staying the same or adding new events. We refine the events send−request by adding two more guards b = TRUE, and etat= valid. And for the events: update_table_same_entry1, update_table_same_entry2 , updare_table−dif we add the guard etat=valid, the route should be valid so it can be updated.

We add three more events route−invalid, route−remove and route−holddown. The first added event presents when the route is considered invalid. In the second one the invalid routes are removed after a flush time. The third event presents when route is considered unreachable

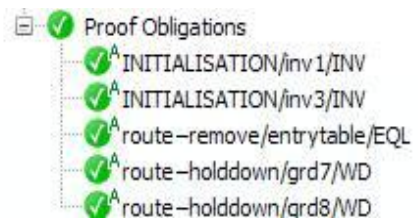
```
    route−invalid:
      ANY
        s ›
        n ›
      WHERE
        grd1:  s ∈ NODE
        grd2:  n ∈ NODE
        grd3:  temps∈ℕ
        grd4:  s ≠ n
        grd5:   etat=valid
        grd6:   temps≥t2
        grd7:   s↦n ∈ entrytable
    THEN
        act1:etat≔invalid
    END

route−remove:
    ANY
        s  , n
    WHERE
        grd1:   s ∈ NODE
        grd2:   n ∈ NODE
        grd3:   s ≠ n
        grd4:   etat=invalid
        grd5:   temps≥t3
        grd6:   s↦n ∈ entrytable
    THEN
        act1:entrytable≔entrytable∪(entrytable∖{s↦n})
    END

route−holddown:
    ANY
        s  ,    n
    WHERE
        grd1:   s ∈ NODE
        grd2:   n ∈ NODE
        grd3:   s ≠ n
        grd4:   temps≥t4
        grd5:   s↦n ∈ entrytable
        grd6:   s∈dom(hopecount)
        grd7:   n∈dom(hopecount(s))
        grd8:   hopecount(s)(n)≥16
    THEN
        act1:etat≔unreachable ›
    END
```

All the proof obligations have been valid automatically or manually in the both model fig1 and fig2. The fig3 shows the number of generated proof obligations rules and the state of each one



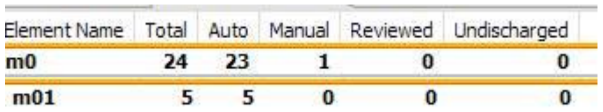
Events
Proof Obligations
INITIALISATION/inv3/INV
INITIALISATION/inv4/INV
INITIALISATION/inv7/INV
INITIALISATION/inv8/INV
INITIALISATION/inv9/INV
INITIALISATION/inv10/INV
send−request/grd6/WD
send−request/grd7/WD
receiv−request/grd6/WD
receiv−request/grd7/WD
update_table−dif/grd8/WD
update_table−dif/grd9/WD
update_table−dif/inv9/INV
update_table_same_entry1/grd8/WD
update_table_same_entry1/grd9/WD
update_table_same_entry1/grd13/WD
update_table_same_entry1/grd15/WD
update_table_same_entry1/grd16/WD
update_table_same_entry1/inv9/INV
update_table_same_entry2/grd8/WD
update_table_same_entry2/grd9/WD
update_table_same_entry2/grd13/WD
update_table_same_entry2/grd15/WD
update_table_same_entry2/grd16/WD

Fig.1 proof obligation rules in initial model



Proof Obligations
INITIALISATION/inv1/INV
INITIALISATION/inv3/INV
route−remove/entrytable/EQL
route−holddown/grd7/WD
route−holddown/grd8/WD

Fig.2 proof obligation rules in the first refinement



Element Name	Total	Auto	Manual	Reviewed	Undischarged
m0	24	23	1	0	0
m01	5	5	0	0	0

Fig.3 proof obligation statistics

V. Conclusion

The formal method has been used to model different concept and answer to different discipline [10], [11], [12]. We have used event-B as a formal method. Thus, we have transformed the informal specifications into a formal notation based on mathematical approach. In this paper, we have modeled the routing information protocol with the event-B method. Going from an initial model and refine it. The Event-B method has assured the correctness of the models and proves it using the proof obligation rules in the platform Rodin.

The proof obligation rules that have been failed are fixed by adding new invariants or strengthen the guards in the events.
In the end we can see that our model in correct and proved using Event-B.

References

[1] B.Boulamaat., A.Amamou, R.Filali,, S.El.mimouni,, M.Bouhdadi,'' Modeling RIP using Event-B'' Proceedings of the 1st International Conference on Mathematical Methods & Computational Techniques in Science & Engineering (MMCTSE 2014)pp152-156

[2] Jean-Raymond Abrial,Modeling in Event-B - System and Software Engineering. Cambridge University Press 2010, ISBN 978-0-521-89556-9.

[3] Edsger W,Dijkstra, Carel S. Scholten, Predicate calculus and program semantics. Texts and monographs in computer science, Springer 1990, ISBN 978-3-540-96957-0, pp. I-X, 1-220

[4] Jean Raymond Abrial, Stefan Hallerstede, Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B Fundamenta Informaticae,Vol.77, No.1-2, 2002, pp. 1-28.

[5] Jean-Raymond Abrial, Michael J. Butler, Stefan Hallerstede, Thai Son Hoang, Farhad Mehta, Laurent Voisin: Rodin: an open toolset for modelling and reasoning in Event-B. STTT, Vol 12, NO.6, 2010, pp. 447-466.

[6] http: //www.event-b.org Rodin Platform

[7] C. Hendrik, RFC 1058, Routing Information Protocol, the Internet Society, June 1988

[8] G. Malkin, RFC 1388, RIP Version 2 - Carrying Additional Information, The Internet Society ,January 1993

[9] G. Malkin ,RFC 2453, RIP Version 2, , The Internet Society ,November 1998

[10] Xin Ben Li, Feng Xia Zhao,Formal development of a washing machine controller by using formal design patterns. Proceedings of the 3rd WSEAS international conference on Computer engineering and applications 2009. ISBN:978-960-474-41-3 ,pp127-132

[11] Štefan Korečko, Branislav Sobota, Csaba Szabó, Using simulation and 3D graphics software to visualize formally developed control systems, Proceedings of the 15th WSEAS international conference on Computers,july 2011,pp 98-103

[12] Adel Ali Ahmed, Norsheila Fisal, Sharifah H. S. Ariffin, Probabilistic real-time routing protocol for mobile ad-hoc networks based on IEEE 802.11b, ACS'11: Proceedings of the 11th WSEAS international conference on Applied computer science,Octobre 2011.

[13] R.-J. Back, Refinement Calculus II: Parallel and Reactive Programs. In: de Bakker J. W., de Roever W. P., Rozenberg G. (eds.), Lecture Notes in Computer Science, Springer, vol 430, pp. 67-93, 1990.

[14] L.Lamport, "The temporal logic of actions," Transactions on Programming Languages and Systems (TOPLAS), vol.16 no.3, pp. 872-923, 1994.