Advance in Electronic and Electric Engineering
ISSN 2231-1297, Volume 3, Number 1 (2013), pp. 91-98

http://www.ripublication.com/aeee.htm

Formal Modeling of Distance Vector Routing Protocol using Event-B

Arun Kumar Singh[1] and Vinod Kumar Singh[2]

[1]PhD Scholar, IET, GBTU, Lucknow, INDIA.
[2]Professor, Electronics Engineering Department, IET, GBTU, Lucknow, INDIA.

Abstract

Ad hoc networks are dynamic networks of mobile hosts with wireless network interfaces without any established infrastructures. Due to dynamic nature of these networks, formal specifications are needed to ensure correctness of routing protocols that are used in these networks. For any routing algorithm route discovery is an important task. Distance Vector is a proactive routing protocol, which continuously maintains the topological information and such route information is available immediately whenever communication is needed.
Formal methods are mathematical techniques which are used to develop software model. Event-B is formal technique that enables user to express the problem at abstract level and then add more details in refinement step to obtain concrete specification. The Event-B model generates proof obligations. For ensuring correctness of the system we need to discharge all proof obligations. In this paper, we have formalized distance vector routing protocol using event B.

Keywords: Formal Methods, Formal Specification, Event-B, Distance Vector, Ad hoc network.

1. Introduction
Ad hoc networks vary from conventional networks in the dynamic topology of interconnections and automatic administration for setting up the network. The topology of the ad hoc can be arbitrary at any time. With the change of the topology of an ad hoc network, the nodes in the network have to update their routing information automatically and instantly [1].

Traditionally, the network routing protocols could be divided into *proactive protocols* and *reactive* protocols. Proactive protocols continuously learn the topology of the network by exchanging topological information among the network nodes. Thus, when there is a need for a route to a destination, such route information is available immediately.

The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm and the **Ford-Fulkerson** algorithm. It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP [2].

Numerous formal methods have been applied to the analysis of network protocols. This includes model checking [3, 4], theorem proving [5], and development by refinement [6, 7]. With respect to routing protocols, probably the most detailed study is that of [8], who used an interactive theorem prover (HOL) together with a model checker (SPIN) to prove different properties of distance vector routing protocols. They carried out case studies analyzing the Routing Information Protocol (RIP) standard and the Ad-Hoc On-Demand Distance Vector (AODV) protocol. In this case study we formally model the distance vector routing protocol using Event-B.

The rest of the paper is organized as follows: Section 2 presents our modeling approach and the introduction of Event-B, Section 3 discusses the formal model of distance vector in Event-B and the section 4 concludes the paper.

2. Event-b AS A Formal Method

A development in Event-B [9] is a set of formal models. A model contains the complete mathematical development of a Discrete Transition System. Event-B has a semantics based on transition systems and simulation between such systems, described in [10]. Event-B models is organized in terms of the two basic constructs: contexts and machines. Contexts specify the static part of a model whereas machines specify the dynamic part. Within the Event B framework, asynchronous systems may be developed and structured using abstract systems [10]. Abstraction can be viewed as a process of simplifying our understanding of a system by identifying the key features of the system to be modeled by focusing the intended purpose of the system and ignoring details of how that purpose is achieved.

Event-B supports refinement to augment the functionality being modeled, or introducing details about the dynamic properties of a model. In refinement steps we refine one model M1 to another model M2, model M2 to model M3 and so on, till the desired functionality is achieved. The states of the abstract machine are related to the states of the concrete machine by gluing invariants J (v, w), where v are the variables of the abstract machine and w are the variables of the concrete machine. With abstraction, we can postpone treatment of some system features to later refinement steps. Event-B provides a notion of consistency of a refinement by generating the proof obligations and providing an environment to discharging the proof obligations while failing proof can help us to identify inconsistencies in a refinement step. Refinement and abstraction allow us to manage the complex system in the design process by first describing the abstract problem, introducing solutions or details in refinement steps to obtain more concrete specifications and verifying that proposed solutions are valid.

2.1 Machines and Contexts

Machines specify behavioral properties of Event-B models. It may contain variables, invariants, theorems, events and variants of a model. Variables v define the state of a machine. They are constrained by invariants I (v). Possible state changes are described by events. While Contexts may contain carrier sets, constants, axioms, and theorems. Carrier sets are similar to types [9]. Axioms constrain carrier sets and constants, whereas theorems express properties derivable from axioms.

Machines and contexts have various relationships: a machine can be "refined" by another one, and a context can be "extended" by another one (no cycles are allowed in both these relationships). Moreover, a machine can "see" one or several contexts [10].

A model can only contain contexts, or only machines, or both. In the first case, the model represents a pure mathematical structure. In the third case, the machines of the model are parameterized by the contexts. Finally, the second case represents a machine which is not parameterized.

2.2 Events

A machine event represents a transition. Each event is made of guard G (v) and action S (v). The guards together denote the necessary condition for the event to be enabled, whereas the actions together represent the way the variables of the corresponding machine are modified. An event can be represented by the term -**when** G (v) **then** S (v) **end**

Events can have no guards, they can be also simple and guarded (keyword where) or parameterized and guarded (keywords any and where). When an event lies in a machine which refines another one then the event may specify (if any) the abstract event(s) it refines (keyword refines). When a refining event refines an abstract event which is parameterized, one may provide some witnesses (keyword with). The status of the event can be normal, convergent or anticipated. A "convergent" event must be a new event in a refined machine (one that does not appear in the abstraction). All convergent events in a machine are concerned with the variant section of that machine. An "anticipated" event is a new event which is not "convergent" yet but should become "convergent" in a subsequent refinement.

3. Formal development of distance vector using event- b

3.1 Informal Description of Distance Vector Protocol

Distance vector routing is the simplest proactive routing protocol. The Distance vector routing protocol [11] operates by having each node *i* maintains a table, which contains a set of distance or cost $\{dij(x)\}$, where *j* is a neighbor of *i*. Node *i* treats the neighbor *k* as the next hop for a data packet destined for node *x*, if $d\ ik = \min \forall j \{d\ ij(x)\}$. The routing table gives the best distance to each destination and which route to get there by periodic updation of information with all its neighbors.

3.2 Formal Development

MACHINE RoutingM

VARIABLES *sent, got, lost, ALinks, nexthop, nodemetrix, chstore, intmednode*

INVARIANTS

inv1: $sent \subseteq MSG$

inv2: $got \subseteq MSG$

inv3: $lost \subseteq MSG$

inv4: $ALinks \in (NODE \leftrightarrow NODE)$

inv5: $got \cup lost \subseteq sent$

inv6: $got \cap lost = \emptyset$

inv7: $nexthop \in NODE \rightarrow (NODE \rightarrow NODE)$

inv8: $nodemetric \in NODE \rightarrow (NODE \rightarrow \mathbb{N})$

inv9: $chstore \in NODE \leftrightarrow MSG$

inv10: $\forall i \cdot i \in NODE \land i \in dom(chstore) \Rightarrow (got \cup lost) \cap chstore[\{i\}] = \emptyset$

inv11: $ran(chstore) \cup (got \cup lost) = sent$

inv12: $\forall i \cdot i \in NODE \Rightarrow chstore[\{i\}] \subseteq sent$

inv13: $\forall m \cdot m \in MSG \land m \notin sent \Rightarrow (m \notin got \land m \notin lost \land (\forall i \cdot i \in NODE \Rightarrow i \mapsto m \notin chstore))$

inv14: $\forall m, i, j \cdot i \mapsto m \in chstore \land j \mapsto m \in chstore \Rightarrow i = j$

inv15: $intmednode \subseteq NODE$

EVENT SEND

ANY s, t, datamsg

WHERE

grd1 : $s \in NODE$

grd2 : $t \in NODE$

grd3 : $s \neq t$

grd4 : $datamsg \in MSG$

grd5 : $datamsg \notin sent$

grd6 : $source(datamsg) = s$

grd7 : $target(datamsg) = t$

THEN

act1 : $sent := sent \cup \{datamsg\}$

END

EVENT RECEIVE

ANY s, t, datamsg

WHERE

grd1 : $s \in NODE$

grd2 : $t \in NODE$

grd3 : $source(datamsg) = s$

grd4 : $target(datamsg) = t$

grd5 : $datamsg \in MSG$

grd6 : $datamsg \in sent \setminus (got \cup lost)$

THEN

act1 : $got := got \cup \{datamsg\}$

END

EVENT ADD_LINK

ANY x, y

WHERE

grd1 : $x \mapsto y \notin ALinks$

grd2 : $x \neq y$

THEN

act1 : $ALinks := ALinks \cup \{x \mapsto y\}$

END

EVENT DEL_LINK

ANY x, y

WHERE

grd1 : $x \mapsto y \in ALinks$

grd2 : $x \neq y$

THEN

act1 : $ALinks := ALinks \setminus \{x \mapsto y\}$

END

EVENT LOSING

ANY s, t, datamsg

WHERE

grd1 : $datamsg \in sent \setminus (got \cup lost)$

grd2 : $source(datamsg) = s \land target(target(datamsg)) = t$

grd3 : $s \mapsto t \notin path(ALinks)$

THEN

act1 : $lost := lost \cup \{datamsg\}$

END

EVENT FORWARD

ANY datamsg, x, y, t

WHERE

grd1 : $t \in NODE$

grd2 : $datamsg \in sent \setminus (got \cup lost)$

grd3 : $x \mapsto y \in ALinks$

grd4 : $target(datamsg) = t$

grd5 : $x \neq target(datamsg)$

grd6 : $x \mapsto datamsg \in chstore$

grd7 : $y \mapsto datamsg \notin chstore$

THEN

act1 : $chstore := (chstore \setminus \{x \mapsto datamsg\}) \cup \{y \mapsto datamsg\}$

act2 : $intmednode := \{x\}$

END

Fig. 1: Variables, Invariants and Events of Machine.

In this paper, we present the formal development of Distance Vector routing protocol using Event B (fig.1).

Routing denotes the selection of paths through a network for sending data from a source to a destination. A path may require the data message to travel over multiple nodes, each node being an intermediate router. In routing protocols each host works as a router and constructs a graph representing the network topology. In this graph, vertices and edges represent routing nodes and direct connection between nodes, respectively. To specify the correct desired properties of protocol at abstract level and in carrying out the development and proofs in subsequent refinement models, is a challenging problem [12]. Section 3.2 describes the formal development of Distance vector protocol.

In the context of machine NODE and MSG are defined as a carrier set which represent the set of nodes and the messages respectively. For the given network it is assumed that there are only finite number of directed nodes. The variables *sent, got* and *lost* are defined as a subset of *MSG (see fig.1.).* The variable *ALinks* which is defined as : ALinks ∈ NODE↔NODE represents the set of active links. The variable *intmednode* represents the intermediate node from where the data message *datamsg* is received. These variables are initialized with null values. There are two more variables *nexthop* and *nodemetric* which gives the information about next node and shortest distance to destination node respectively. The invariant *inv10* which is written as : $\forall i \cdot i \in NODE \land i \in dom(chstore) \Rightarrow (got \cup lost) \cap chstore[\{i\}]$ ensures that the massage is either received , lost or it is in the transit. The invariant *inv11*is expressed as: $ran(chstore) \cup (got \cup lost)$, ensures that total send messages is equal to total distributed messages (received, lost, transit) in the network.

The event *SEND* models the transfer of fresh data message *datamsg* from source node (s) to destination node(t).The guard *grd5*ensures that datemessage has not been sent previously. The event *RECEIVE,* denotes the receiving of data message *datamsg* by destination node(t). The guard grd6 ensures that data message *datamsg* neither received nor lost. It will be eventually received by destination node (*t*). The event *ADD_LINK* formalize the linking of new nodes while event *DEL_LINK* represents the deletion of existing link between nodes. The event *LOSING*, represents those data message *datamsg* who have sent by source node (s) but not received by destination node(t). The guard grd3 ensures that there is no path between source *s* and destination node *t*. The forward event models the forwarding of data message *datamsg* to their neighboring nodes. The action *act1*of this event make the entry for node to which data message is forwarded.

Each time when a data message *datamsg* is received by node *x* from node *y*, the node *x* updates its routing table if following condition holds true:

$$nodemetric(x)(p) > (nodemetric(x)(y)+nodemetric(y)(p))$$

Where nodemetric(m)(n) is the distance from node m to n and p, m, n : NODE. The following action change the routing table of node x:

$$nodemetric(x) \coloneqq nodemetric(x) \;\; + \{p \mapsto (nodemetric(x)(y)+nodemetric(y)(p))$$

The next hop entry is also updated through the action: *nexthop(x)*$:=$ *nexthop(x)*+$\{p \mapsto y\}$.

4. Conclusion

This paper presents formal modeling of Distance Vector routing protocol. The work regarding formal modeling of topology discovery in changing environment can be found in [13] but our work is a significant case study which also implements the updation of routing table using shortest distance between nodes. We have used Event-B as a formal method for writing specifications. Event-B generates proof obligations which is need to be discharged for ensuring correctness of the system. Rodin platform is used for discharging proof obligations. Total 31 proof obligations generated by system. All of them are discharged automatically. The advantage of Distance Vector is its simplicity but the tendencies of creating routing loops make it unsuitable for ad hoc networks. The Distance Vector can be further refined by adding the destination sequence number which helps the mobile nodes to distinguish stale route information from the new and thus prevent the formation of routing loops.

References

[1] Guoyou He,DSDV Guoyou He., Destination-sequenced distance vector (DSDV) protocol. Technical report, Helsinki University of Technology, Finland.

[2] Tanenbaum Andrew S.: Computer Networks,Prentice Hall, Inc., 1996, ISBN 7-302-02410-3.

[3] Christel Baier and Joost-Pieter Katoen. Principles of Model Checking. The MIT Press, 2008.

[4] Gerhard J. Holzmann. The Spin Model Checker: Primer and Reference Manual. Addison–Wesley, 2003.

[5] Marco Devillers, David Griffioen, Judi Romijn, and Frits Vaandrager. Verification of a leader election protocol: Formal methods applied to ieee 1394. Form. Methods Syst. Des.,16(3):307–320, 2000.

[6] Abrial, J.R., J.R., Cansell, D.,M´ery, D.:A Mechanically Proved and Incremental Development of IEEE 1394 Tree Identify Protocol. Formal Asp. Comput. 14(3), 215–227,2003.

[7] A Udaya Shankar and Simon S Lam. A stepwise refinement heuristic for protocol construction.ACM Transactions on Programming Languages and Systems, 14(3):417–461, 1992.

[8] Karthikeyan Bhargavan, Davor Obradovic, and Carl A. Gunter. Formal verification of standards for distance vector routing protocols. J. ACM, 49(4):538–576, 2002.

[9] Abrial, J.R.:Modeling in Event-B: System and Software Design. Cambridge University Press, 2010.
[10] Abrial, J.R.: Extending B without Changing it (for developing distributed systems). Proc. of the 1st Conf. on the B method, H. Habrias (editor), France, pages 169–190, 1996.
[11] Perkins Charles E., Bhagwat Pravin: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, London England UK, *ACM SIGCOMM'94*, 1994, pp. 234–244.
[12] DominiqueM´ery and Neeraj Kumar Singh:Analysis of DSR Protocol in Event-B 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (2011) 401-415.
[13] Hoang, T.S., Kuruma, H., Basin, D.A., Abrial, J.R.: Developing Topology Discovery in Event-B. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 1–19.Springer, Heidelberg (2009)