Formal modelling of a robust Wireless Sensor Network routing protocol

Kashif Saghar [*] William Henderson[*] David Kendall[*] Ahmed Bouridane[*†]
[*] School of Computing, Engineering and Information Sciences, Northumbria University,
Pandon Building, Newcastle upon Tyne NE2 1XE, United Kingdom
[†] College of Computer and Information Sciences, King Saud University
P.O.Box 51178 Riyadh 11543, Kingdom of Saudi Arabia
kashif.saghar@unn.ac.uk

Abstract

Because of their low cost, small size, low resources and self-organizing nature a Wireless Sensor Network (WSN) is a potential solution in hostile environments including military applications. However, the broadcasting nature of radio transmission; their limited computing, power and communication resources; unattended and potentially hostile nature of the environment they operate in make WSNs prone to Denial of Service (DoS) attacks. Although many schemes have been proposed to address DoS attacks their effectiveness is yet to be proven. The traditional methods used (i.e. visual inspection, computer simulations and hardware implementations) can only detect errors but cannot verify that the whole system is error free. Therefore, new techniques to automatically determine the worst cases and hidden errors in WSNs are much desired. After an initial investigation using a formal verification which clearly shows that Arrive routing protocol is vulnerable to different DoS attacks, this paper proposes a method for its security. The finding contradicts the claim of the developers of Arrive that it is immune to black hole attacks. Several other DoS attacks were also found to be successful in Arrive routing protocol. The formal model generates the trace to confirm how an attack is possible in the protocol. However, it was found that INA attacks are addressed by Arrive protocol. To our best knowledge the results discussed in this paper have not been presented, proved or published before.

1. Introduction

A WSN is composed of small embedded computers (called nodes) that communicate wirelessly to perform a particular task. Because of their low cost, small size, low resources and self-organizing nature a WSN is a potential solution in hostile environments and military applications.

A WSN is an *adaptive embedded system* because the system as a whole tries to maintain and improve its performance in the presence of uncertain/hostile environmental conditions as well as failure/destruction of some of the nodes. WSNs can be deployed in large numbers in hostile conditions and the destruction of some WSN nodes can still achieve the desired functionality. The aim of a WSN is to gather data from remote locations (possibly hostile) and route it wirelessly to secure places (i.e. base station). A node that gathers the data is called 'Source' and the nodes intersted in data are called 'Sinks'. Unlike in a normal network each node can act as a router. The broadcast nature of radio transmission; limited computing, power and communication resources; unattended and potentially hostile nature of the environment are all major challenges which have to be overcome in the process of data routing.

Attackers in hostile environments can launch different DoS attacks to prevent data from reaching the base station. As in the case of our recent publication [10], we use the term *Denial of Service* in a general sense to mean an adverse effect of any malicious external agent (attacker) on the correct or timely delivery of data from source nodes to the sink nodes. Again the particular focus is on the effects of denial of service attacks on routing protocols in WSNs. A brief summary of attacks that were considered has already been presented in [10] and more detailed surveys of attacks are available in [8, 13].

Many secure solutions for routing protocols have been proposed to overcome the DoS attacks and to adapt to hostile environments. These schemes are basically adaptive software tools since they attempt to improve data routing in the presence of faults (attacks) and degrading networks (i.e., when some nodes become malicious). However, the credibility of the secure schemes in the presence of attacks is yet to be proven. The traditional methods used such visual inspection, computer simulations and hardware implementations can only detect a few errors but cannot guaranty the whole system to be error free. These methods can only verify the presence of a fault but not its absence. We propose in this paper a new technique based on 'formal modeling' which can automatically check the worst cases and hidden errors. Formal modeling can verify the system against every single execution trace and thus can confirm that the whole system functions as expected. The paper involves applying this approach to check WSNs protocols rigorously against DoS attacks.

A modeling framework, which was developed by the authors [10], allows for an automatic analysis and vulnerability of routing protocols to different DoS attacks using formal modeling and theorems (LTL properties). This paper is an extension of the previous work. The formal model confirms that Arrive [7], seen as a robust protocol, is indeed vulnerable to some DoS attacks. However, in this paper UPPAAL [2] is used instead of Spin [5] for the modeling the protocol. This is carried out to overcome the state space explosion problem when Spin is used to verify protocols for bigger networks (¿5). The problem arises because of the development of a separate channel model in Spin, which can only provide point to point communication. UPPAAL not only possesses all the properties of Spin but also provides broadcast communication highly suited to WSNs. This reduces the model size considerably thus allowing us to check additional number of nodes without the state space explosion problem.

The rest of this paper is organized as follows: Section 2 briefly discusses the related works while a description of Arrive protocol is given in Section 3. The proposed formal framework including the results obtained are described in Section 4. Conclusions and further work are presented in Section 5.

2. Related Work

It is accepted within the research community that computer simulation is often inadequate for finding errors in routing protocols. The development of formal models to determine various aspects of routing protocols is becoming an important issue. Formal modeling has also been used in the analysis of security protocols and different hidden attacks have been discovered using formal modeling. For example, [12] discovered faults in TinySec and LEAP protocols; the formal analysis in [11] showed successful attacks in SNEP (SPINS) [9]; the work on [4] developed a new approach (Sledge method) based on Spin [5] and verified μTESLA [9] and LEAP [14] using this method. The most noteworthy work is presented in [1] where the authors used Spin [5] to analyze the effect of some DoS attacks on ad-hoc routing protocols based on Dynamic Source Routing (*Ardiadne* and *endairA*). The authors in [1] have used an automated security evaluation process and analyzed all topologies for networks of up to 5 nodes. This work was extended by us in [10] to check WSN protocols rigorously against DoS attacks and a modeling framework was developed. This framework adopted for the modeling and analysis of a simple routing protocol (TinyOS Beaconing [8]) and its robust version (with μTESLA [9]) using Spin. The formal framework also revealed unknown flaws in the Rumor Routing protocol [3] and Directed Diffusion [6] protocols.

3. Arrive Routing Protocol

Arrive protocol [7] has been developed to provide robustness against malicious and failed node attacks by maintaining neighbor reputation. Unlike other routing protocols, the reputation is maintained locally and is not shared with other nodes. Therefore, the nodes cannot misrepresent their data. It uses randomness and probabilities to reinforce the reliable neighbors. Also packets are routed in multiple paths to prevent a single failure affecting data to reach the sinks. The nodes not only forward data designated to them to reputed nodes (Direct Participation) but also eavesdrop the traffic. In the case of a message failure in neighborhood (a neighbor does not forward message after receiving a message from another neighbor), the node forwards the message that is not addressed to it (Passive Participation). Although this method induces extra traffic, it provides robustness.

Arrive protocol uses first a breadth technique to assign nodes level like the TinyOS beaconing protocol [8]. The base station has a level of 0, the nodes one hop away from the base stations have the level of one and so on. The authors have not explained this issue in detail but it is expected that a message of the following format is initiated and broadcast by the base station:

$$BS \rightarrow * : (level, ID_{BS}, L)$$

Here 'level' is the message type, followed by the sender ID and the level L attached. The base station initiates with the level 0. A node upon receiving a lower level updates its current level and rebroadcast the level message with the new level L:

$$N \rightarrow * : (level, ID_N, L)$$

Moreover, if a node receives a lower level beacon it will mark the sender as its parent. On the other hand, senders with the same lev are assigned as neighbors. The sender nodes with higher levels are ignored. Once the setup phase is completed a source node S unicast its data to either parent or neighbor depending upon the node's reputation. A node with a lower reputation is not selected. The message format is:

$$S \rightarrow N : (level, ID_N, [ID_S, Event_E)$$

The message contains the source ID and event number E. A node N upon receiving this message again forwards it to either parent or neighbor:

$$N \rightarrow P : (level, ID_P, [ID_S, Event_E)$$

The node sometimes acts as a passive forwarder by forwarding the data not addressed to it. This is a probabilistic process and requires a condition. When receiving data, a node within the radio range can either forward it farther or to a node outside the current node's radio range. The developers of Arrive, however, warned that the 'Passive participation' must be used with caution because unidirectional links and hidden terminal problems may cause the passively participating nodes to act erroneously.

4. Formal Model

It is believed that DoS attacks such as black hole, hello flood, sinkhole, spoofing and wormhole attacks are still possible in Arrive protocol and can lead to a significant data loss. One of the aims of this paper is to apply formal modeling on Arrive, using UPPAAL [2], to verify whether it is vulnerable to DoS attacks. When modeling Arrive protocol, it is assumed that the channel is ideal i.e. no message is lost because of collision or noise; the nodes are placed in rectangular grid and have equal radio range; and that the node density of network is limited to 4. In formal modeling one has to pay the price of high resources (computation, memory etc) so the assumptions are applied to simplify the model.

This section is further organized as follows: Subsection 4.1 briefly describes the formal model of Arrive; Subsection 4.2 describes the theorems or LTL properties applied on Arrive; Subsection 4.3, 4.4, 4.6, 4.5 and 4.7 describe detection of different DoS attacks by the formal framework.

4.1. Proposed Model

The complete UPPAAL model is split into 3 parts: the Node model, the Event Generator (EG) model and the Sink model. The ***Node model*** starts from INIT_PHASE which indicates the setup phase. A node waits for a level message and upon its reception it is updated. It is worth noting that all the node levels are initialized to a very high value and gets updated only if a lower level value is received. There is a flag 'LevelRec' in the model which is set when a level beacon is received and is cleared when the level beacon is broadcast (SEND_LEVEL). A variable BusyNodes indicates how many nodes are busy receiving or sending messages. The EG model also uses this variable and moves to the next phase when this variable becomes 0.

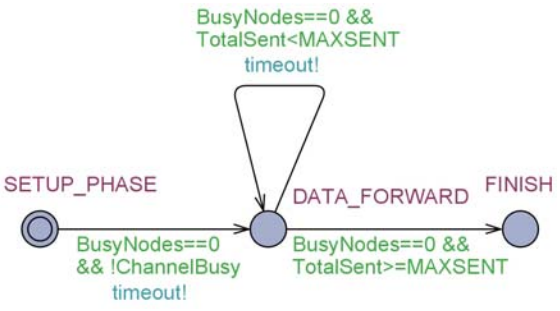


Figure 1. Event Generator model for Arrive Protocol

All Node models leave the setup phase and move to data forward phase when the EG model triggers them using the 'timeout' message. This broadcast message is used to indicate an event to all nodes simultaneously. All nodes initialize their parents and neighbors using InitializeParentOrNeighbor() function which utilizes the topology matrix and the levels received in the setup phase. If a node is a source node, it senses data (SENSE_DATA) and broadcast (SEND_DATA) it. Another function SelectParentOrNeighbor() is used to select a neighbor before data is forwarded.

If a node is not the source, it will move to LISTEN mode and will expect data messages. Upon receiving data addressed to it (REC_DATA_DIRECT) a legitimate node forwards it. If data is not addressed to a legitimate node (Msg_Level!=NID), it checks if the data has been addressed to its neighbor (Topology[NID][Msg_Level]). If so, the node moves to passive forwarding phase (PASSIVE_FORWARD) and remains in this phase for a certain time (10 clock cycles). The model checks two conditions here: (i) Data has been forwarded by the observed node during this time and (ii) Data has been forwarded to a neighbor not within the range of current node i.e. Topology[NID][Msg_Level] is false. If any of the above conditions is not satisfied and a certain time has elapsed (DELAY), the Node model checks the probability to forward data (PROBIBLITY_CHECK) using a global function ProbabilityCheck(). If the probability is higher than the threshold value, the Node model will forward data (FORWARD_DATA) otherwise it moves back to the LISTEN location. Another local variable 'Sender' is used to track sender ID of the data message in order to ensure that the data is not sent back to it.

The ***Event Generator model*** is shown in Figure 1. The EG model generates different events required by the protocol. The model starts with the SETUP_PHASE. After all Node models complete the setup phase (BusyNodes becomes 0), the nodes are triggered to move to the operation phase by sending their corresponding timeout to all

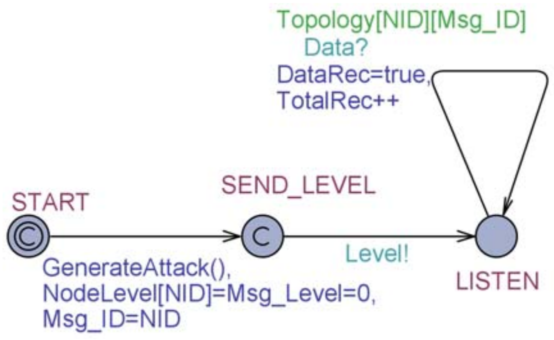


Figure 2. Sink model for Arrive Protocol

Node models. This message also triggers the source to initiate the data message. The model tracks the data flow and when all nodes become idle (BusyNodes is 0), it generates a new timeout message to enable the source node to generate a new data message. A variable 'TotalSent' is incremented each time the source generates a new message and thus tracks the total number of messages generated by the source. When all messages have been sent, the EG model moves to the FINISH location. The maximum number of data messages generated is limited by using MAXSENT constant to reduce the state space.

The ***Sink model*** is shown in Figure 2. The sink model immediately moves out of the initial state (START). The function MakeWormhole() is used to generate a wormhole tunnel if needed. The Sink model then broadcast the level message (SEND_LEVEL) with level 0. Once this done, the model remains in LISTEN state with a self loop. This loop is only triggered if a node within the radio range broadcast data. The flag DataRec is set even if a single data message is received. A global variable DataRec keeps track of the total data messages received by the sink.

4.2. Verification

When the model was constructed, simple theorems (LTL properties) were applied to confirm that Arrive achieves the desired operations. The theorems applied perform the sanity checks and confirm that the safety & liveliness properties hold in the protocol. The liveliness properties are defined as 'something good will eventually happen' while the safety properties are defined as 'nothing bad will never happen' [2]. Thus data which never reaches the sink can be termed as safety violation. Incorrect level assignment is termed as liveliness failure. Note that in UPPAAL *E* and *A* indicate 'eventually' and 'always', respectively, while <> and [] are symbols for 'one case' and 'all cases' respectively. The theorems/properties checked are:

- The sanity checks that all the nodes eventually broadcast a level message, where N is ID of a node. This property is verified for all nodes in the network.

$$E <> Node_N.SEND\_LEVEL$$

- The sanity checks that a source node eventually senses and broadcast data at least once, where S is ID of source node:

$$E <> Node_S.SENSE\_DATA$$

- The sanity checks that the protocol eventually will enter 'Data Forward Phase'. This also confirms that the protocol will always finish the setup phase.

$$E <> Protocol.DATA\_FORWARD(true)$$

$$A[]Protocol.SETUP\_PHASE(false)$$

- The final sanity test verifies that the protocol will not deadlock and will finish eventually i.e. it goes to FINISH state. It can also be treated as liveliness property. Note that UPPAAL model, however, does deadlock, when EG is in FINISH location, as only a limited number of messages are sent. If the guard to transmit a certain number of messages is removed from the EG model, then the system will never move to deadlock state and will keep on sending messages for ever. However this has to be prevented, since our main aim is to check certain properties and prevent the state space explosion problem. To address this problem, only a finite number of traces are checked. A protocol only goes to FINISH state if the source node has sent the desired number of data messages (we set it to be 4) and no node is in data forwarding mode. Either of the two properties can perform this check:

$$E <> Protocol.FINISH$$

$$A[]!Protocol.FINISH$$

- The liveliness checks that all nodes get the correct level when protocol is in operation phase:

$$Protocol.DATA\_FORWARD \Rightarrow (NodeLevel[N] == ExpectedLevel[N])$$

- The liveliness property verifies that a source node S has at least one of its parent or neighbor attaining a lower or equal level so that the source will send the data to this node after the setup phase. The Arrive protocol allows to send data to only a parent (low level) or neighbor (same level).

$$Protocol.DATA\_FORWARD \Rightarrow ((NodeLevel[N1] <= NodeLevel[S]) \; OR(NodeLevel[N2] <= NodeLevel[S]))$$

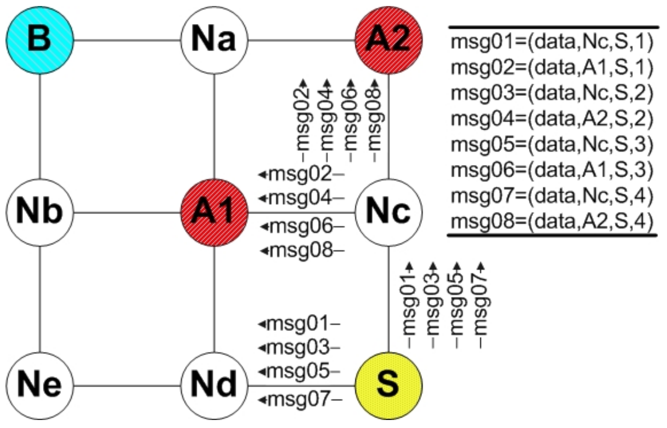


Figure 3. Trace of UPPAAL showing that Black hole is possible in Arrive

- The safety property checks that the base station will eventually receive at least one data from the source.

$$E <> Sink.DataRec$$

- The safety property check that when the protocol completes the number of received data messages at the sink or base station are more than N, where N≤TotalSent.

$$Protocol.FINISH \Rightarrow (TotalRec > N)$$

This final theorem/property is most important and our model confirms that N is reduced to 0 even in the presence of 2 malicious nodes located 2 hops away from the source node. Next, we have applied different attack models to the Arrive protocol to confirm whether the properties/theorems were still valid in the presence of these attacks. In the case of any safety and liveliness property failing, Arrive protocol is considered susceptible to that particular attack and UPPAAL automatically generates a trace. This trace shows how an attacker has succeeded. In the next sections we will discuss these attacks and the extent of their successes or failures.

4.3. Black hole Attack with a single path

In a *black hole* attack, a malicious node joins the network and then either discards all the messages it receives or performs selective forwarding. It was confirmed that Arrive protocol has successfully defeated a single black hole attack. However, a problem arises when there are 2 black hole attacks within any node's range especially when there is no legitimate parent/neighbour of that node. Figure 3 shows a trace generated by UPPAAL in which the safety property fails in the presence of 2 black hole nodes. As the level of propagation was completed without any problem so it has not been included in the trace. The message format used in the trace is (Type,ID,Source,EventNo) where Type is the message type(Level/Data); ID is the destination ID; Source is the source ID and EventNo is the unique Event ID for each event.

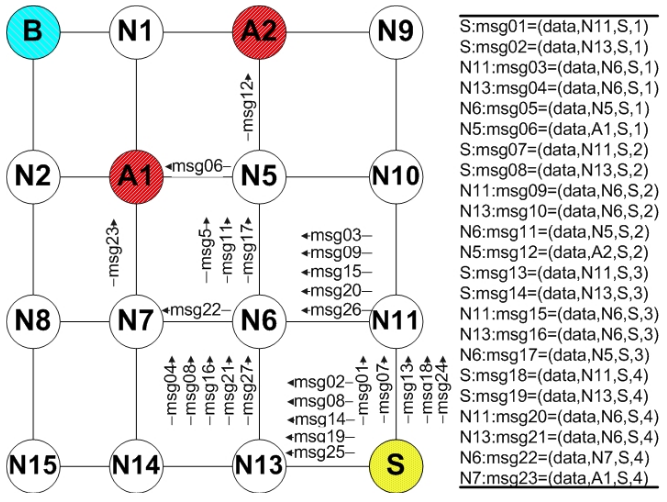
Note that the 2 black holes are not even colluded (within one another's range). The source node keeps on sending data to node Nc as it always eavesdrops if the node Nc has forwarded the data. This also improves the ranking of Nc. The node Nd will not forward the data passively as node Nd has not eavesdropped data sent to any of its neighbors (Nc and Nd are not the neighbors). The node Nc first attempts to forward data to node A1. A1 does not forward data, so the ranking is lowered and at the next time Nc selects A2 for data forwarding. Unfortunately A2 is also an attacker node so the data is lost again. Nc lowers the ranking of A2 and tries to transmit data to A1 and this process continues. Note that even the presence of another node having a lower level than node Nc will not help here. The reason being that the nodes only forward data to their neighbors (same level nodes) or parents (lower level nodes). We later confirmed this by employing 4x4 and 5x5 grids instead of 3x3 grid. Thus a flaw in Arrive protocol was detected through the use of our framework. Even if nodes A1 and A2 are dead, the same error is still possible. Note that a route does exist between S and B via S-Nd-Ne-Nb-B. Any other node's message forwarded by node S (higher level nodes in a larger network) will be lost as well because this will also follow the same path. Even if node S sends half the packets through node Nd, the other half that goes through Nc will be lost and Arrive cannot detect it.

4.4. Black hole Attack with multiple paths

In section 4.3 we discussed that if Arrive uses 2 or more disjointed paths to transmit the same data, the data may eventually reach the base station. To confirm this finding and to model the disjoint multipath technique used in Arrive, a larger network of 16 nodes was considered. The Node model presented earlier was modified so that the source nodes can unicast data to multiple paths.

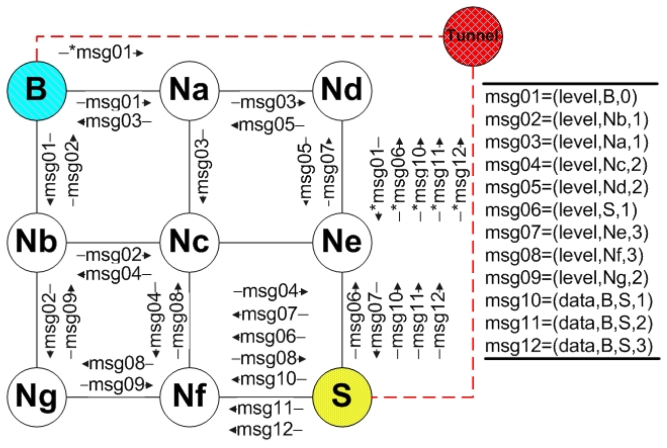
The safety and liveliness properties are violated again as shown by the trace of Figure 4. The message format (Type,ID,Source,EventNo) in the trace is amended this time and the Sender Node's ID is added at the beginning of the message. This has been done to make the trace more readable. Moreover, to simplify the process the level beacons are omitted from the trace and only those messages are placed in the trace which are directly received.

In Arrive protocol the node only stops forwarding to a neighbour/parent if the reputation or ranking is below a threshold value which was set to 1. Therefore, a node will



```
S:msg01=(data,N11,S,1)
S:msg02=(data,N13,S,1)
N11:msg03=(data,N6,S,1)
N13:msg04=(data,N6,S,1)
N6:msg05=(data,N5,S,1)
N5:msg06=(data,A1,S,1)
S:msg07=(data,N11,S,2)
S:msg08=(data,N13,S,2)
N11:msg09=(data,N6,S,2)
N13:msg10=(data,N6,S,2)
N6:msg11=(data,N5,S,2)
N5:msg12=(data,A2,S,2)
S:msg13=(data,N11,S,3)
S:msg14=(data,N13,S,3)
N11:msg15=(data,N6,S,3)
N13:msg16=(data,N6,S,3)
N6:msg17=(data,N5,S,3)
S:msg18=(data,N11,S,4)
S:msg19=(data,N13,S,4)
N11:msg20=(data,N6,S,4)
N13:msg21=(data,N6,S,4)
N6:msg22=(data,N7,S,4)
N7:msg23=(data,A1,S,4)
```

Figure 4. Trace of UPPAAL showing that Black hole is possible in Arrive even using multiple paths



```
msg01=(level,B,0)
msg02=(level,Nb,1)
msg03=(level,Na,1)
msg04=(level,Nc,2)
msg05=(level,Nd,2)
msg06=(level,S,1)
msg07=(level,Ne,3)
msg08=(level,Nf,3)
msg09=(level,Ng,2)
msg10=(data,B,S,1)
msg11=(data,B,S,2)
msg12=(data,B,S,3)
```

Figure 5. Trace of UPPAAL showing that Wormhole is possible in Arrive

not select a neighbour/parent even if it drops a single packet. Still the safety property fails for N up to 4. The trace confirms the path taken by the nodes. Although the data is sent to 2 disjoint nodes N11 and N13, but both the data packets are eventually lost. It is worth noting that this is a low density network with only 2 disjoint paths possible from the source.

An interesting point to note here is that the nodes use reputation in the model after a single message loss. However, this is not actually the case in a real scenario. So one may ask the question of how many packets may be lost before the black hole nodes are isolated. Moreover, this is a small network and for larger networks, more routes can occur through the black holes resulting in further loss of data. As explained in Section 4.3 passive forwarding is not efficient here.

4.5. INA

In Invisible Node Attack (INA), the attacker remains invisible and retransmits the routing information without adding itself and the data packets are later dropped. It was confirmed that INA is unsuccessful in Arrive. Although the nodes gain incorrect level (liveliness property fails), but, the safety properties do hold thus confirming that the data is received by the sink. This is because INA would enable a node to get a level lower than the actual one. The only side effect is that parents may become neighbors, while neighbors are removed. Moreover, INA creates virtual parents. So when a node forwards data to its virtual parent (2 levels away) the nodes which are one level away from the sender node can perform passive forwarding so that the data is eventually sent. The ranking of the virtual parent is dropped resulting in INA not forwarding data. The developers of Arrive were unaware of this fact though they never claimed it. But our model checker has proven that passive forwarding fails in INA.

4.6. Wormhole Attack

In a *wormhole* attack, an attacker records the packets (or bits) at one location in the network, tunnels these to another location and then retransmits into the network. Therefore, a wormhole attack not only disturbs the correct routing but is also the precursor to many other attacks such as black hole, sink hole, etc. Figure5 shows a trace generated by UPPAAL in which both the safety and liveliness properties fail in the presence of a wormhole attack. The message format in the trace is again reduced to (Type,ID,Source,EventNo) where *Type* is message type (Level/Data); *ID* is the destination ID; *Source* is the Source ID and *EventNo* is the unique Event ID for each event. Here a wormhole tunnel exists between node B and S. Thus node S gets a level of 1 and the nodes which should be its parents (Ne and Nf) get a level 2 which is higher than that of node S. The messages msg01 to msg09 explain the setup phase and how the nodes acquire the levels. When the source node S sends data, it has only one parent node B which is not within its radio range. Therefore, whenever a node transmits data to B the node ranking of B is lowered. But as there is no other node (neighbour/parent) that can be chosen to be the next hop, the data is always sent to B and thus lost. Moreover, nodes Ne and Nf cannot act as passive forwarders. The reason is that the necessary condition for a passive forwarding (i.e. B is within their range) is nonexistent. This clearly shows that the wormhole attack was successful in Arrive protocol.

4.7. Other DoS Attacks

Our framework has confirmed that Arrive protocol is susceptible to many other DoS attacks such as hello flood, rushing, sink hole, spoofing attacks etc. The *hello flood* attack involves the use of a high power transmitter (or powerful laptop) by an attacker to broadcast the routing or other information with the purpose to convince every node within its radio range that the attacker is a genuine neighbor node. Our formal framework confirmed that if the hello flood attacker's transmissions reach all the network nodes (unidirectional), then the hello flood eventually fails in Arrive protocol. Since all nodes have the same level (liveliness property fails) the passive forwarding will enable the data to eventually reach the sink. However, the probability of passive forwarding should be low so that hello flood causes considerable message loss before being detected as a malicious parent. Our formal model confirms that, if all the nodes are not within the attacker's radio range, the hello flood is successful in Arrive protocol. In many cases, a subset or only a few nodes will receive these unidirectional transmissions and our formal framework has also confirmed that the nodes get incorrect level (liveliness property fails) and data will not reach the sink (safety property fails).

A *spoofing* attack refers to altering or replaying the routing information. This can lead to the creation of inaccurate or unstable routes. A *sink hole* is a potent form of spoofing in which the attacker makes itself particularly attractive to all nodes within its radio range, usually by advertising low cost routes to all destinations. Our framework has confirmed that both these attacks are successful in Arrive protocol especially when the attacker node behaves as if it is a base station by sending a level of 0. Even if the attacker is within the radio range of other nodes, passive forwarding will not be adopted by any node, assuming that this node is the base station. The proposed formal framework has also confirmed that spoofing attacks are successful since any node can spoof level of 0 and thus can disrupt the whole network.

In *rushing attack*, an attacker transmits incorrect messages, prior to the legitimate ones, with the aim that the nodes reject the legitimate messages and accept the incorrect/false messages. The successful wormhole, sinkhole and hello flood attacks in Arrive protocol have automatically made possible rushing attacks. The nodes accept incorrect messages in level propagation and reject the correct level beacons. Moreover, liveliness property violation will always mean that Rushing attack is successful.

5. Conclusions

Although Arrive protocol provides robustness against malicious and failed nodes by maintaining neighbor reputation, the developers of the protocol themselves suspect that there will be abundance of additional messages in using passive participation due to hidden terminal problems. Also, unidirectional links are the other cause of concern for them. Finally, Arrive protocol also assigns a level to each node based on hop distance from the base station. However, Arrive developers were aware that malicious nodes may lie about their level or replicate these levels to draw more traffic (spoofing). In addition, attackers may forward packets into oblivion to increase their reputation (sinkhole).

Despite these problems, Arrive is still vulnerable to black hole attacks when a node in the network has only black hole parents and neighbors. A black hole is also not detected by off-spring nodes which have forwarded data to that node. This leads to all the down stream traffic being lost. This problem has been detected by our formal framework which has also confirmed that the wormhole will assign incorrect levels to nodes enabling the data to be sent to oblivion. Because the wormhole is usually composed of a multi-hops tunnel, the passive forwarding also will not be possible since a tunnelled node will not be within the radio range of the nodes. In addition, a hello flood attack is also possible in Arrive. However, the problem of INA has been addressed using passive forwarding as conformed by our formal model. Finally, Arrive protocol is susceptible to sinkhole and spoofing attacks as indicated by the developers of Arrive themselves and were also confirmed by our formal framework. Hence, we have shown that by using formal modeling hidden bugs in any routing protocol can be detected automatically. Our future work involves testing more secure and robust routing protocols using formal modeling.

References

[1] T. Andel and A.Yasinsac. Automated evaluation of secure route discovery in MANET protocols. In K. Havelund, R. Majumdar, and J. Palsberg, editors, *Proceedings of 15th International SPIN Workshop on Model Checking Software (SPIN 2008),Los Angeles, CA, USA*, volume 5156 of *Lecture Notes in Computer Science*, pages 26–41. Springer, August 10-12, 2008.

[2] G. Behrmann, A. David, K. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST) 2006*, IEEE Computer Society, pages 125–126, 2006.

[3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA*, pages 22–31, October 2002.

[4] Y. Hanna, H. Rajan, and W. Zhang. Slede: a domain-specific verification framework for sensor network security protocol implementations. In *Proceedings of the first ACM conference on Wireless network security (WISEC 2008), Alexandria, VA, USA*, pages 109–118, 2008.

[5] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99) , Seattle, Washington, USA*, pages 174–185, August 1999.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[7] C. Karlof, Y. Li, and J. Polastre. Arrive: Algorithm for robust routing in volatile environments. In *Technical Report UCBCSD-02-1233, Computer Science Department, University of California at Berkeley*, March 2003.

[8] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, volume 1, pages 293–315, September 2003.

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. SPINS: Security Protocols for Sensor Networks. In *ACM Mobile Computing and Networking*, volume 8, pages 521–534, Sep 2002.

[10] K. Saghar, W. Henderson, and D. Kendel. Formal modelling and analysis of routing protocol security in wireless sensor networks. In *PGNET 2009,Liverpool, UK*, pages 179–184, 22-23 June 2009.

[11] L. Tobarra, D. Cazorla, and F. Cuartero. Formal Analysis of Sensor Network Encryption Protocol (SNEP). In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS 2007), Piscataway, NJ, USA*, pages 767–772, Pisa (Italy), October 2007.

[12] L. Tobarra, D. Cazorla, F. Cuartero, G. Diaz, and E. Cambronero. Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP. In *Proc. of the First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07)*, pages 95–106, Albacete (Spain), September, 2007. IFIP Main Series, Springer.

[13] A. Wood and J. Stankovic. Denial of service in sensor networks. In *IEEE Computer*, volume 35, pages 54–62, Sep 2002.

[14] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *ACM Conference on Computer and Communications Security (CCS'03)*, pages 62–72, October 2003.