# FSM Resentment Using MAC Cross Layer Attacks

**1st Surbhi Agarwal**
Computer Science Engineering
Jaipur National University
Jaipur,India
surbhiagarwal2k19@jaipur.ac.in

**2nd Prof(Dr.)Harvir Singh**
Computer Science Engineering
Jaipur National University
Jaipur,India

**3rd Jameel Ahmad Qureshi**
Computer Science Engineering
Jaipur National University
Jaipur,India
jameelqureshi41@gmail.com

***Article Info***
***Volume 84***
***Page Number: 263 – 277***
***Publication Issue:***
***January/April 2021***

***Article History***
***Article Received**: 04 October 2020*
***Revised**: 14 November 2020*
***Accepted**: 22 December 2020*
***Publication**: 31 January 2021*

**Abstract**— Attacks that cause a denial of service (DoS) are difficult to avoid and defend against. The focus of this study is on DoS attacks in wireless ad hoc networks that propagate from the MAC to the routing layer, causing essential routes to be broken. We show numerous traffic patterns that a clever attacker might create to cause a Denial of Service attack in one or more ad hoc network nodes. We concentrate on the features of the IEEE 802.11 MAC protocol and attack propagation to the routing layer in particular. We're particularly interested in attacks that leverage low-rate traffic patterns to disable one or more specific nodes or partition the network. We propose an attack detection system based on Extended Finite State Machines (EFSM) modelling of MAC protocols and present a broad design for an Intrusion Detection System that can generate attack patterns and assess the authenticity of communication patterns in the network.

***Keywords****—MAC,EFSM,IEEE 802.11,DoS*

## **Introduction**:

in this paper the function and behavior of a finite state machine (fsm) which mainly working for identify the malicious node in mobile ad hoc network due to mobility nature it become easy for attacker to send distrusted message or false message in network and try to create confusion between sender and receiver in case of packet forwarding so fsm technique used for detecting and solved this problem. the fsm architecture has discuss, firstly, the designs are defined as a five tuples (q, q0, n, δ, f), where q is the set of all possible states, q0 is the initial state, n is the set of node operations, δ is a function that maps node operations from a previous state to the current state and f is the set of final states that correspond to malicious behaviors in network.

### Design of Detection Engine

By monitoring the activities at 802.11 Protocol the engine performs and comparing them with the predefined set of design of model. For this implementation we use the network interface card at driver level. Which can be performed by utilizing a number of Application Programming Interfaces like Network Driver Interface Specification interface depending on platform the engine designed? In sequence to present them we use a Finite State Machine (FSM). In the remainder of the section we exemplify the design that describes the correct operation of the 802.11 MAC protocol. In order to simplify the presentation of the detection engine the detailed description of design divided into three set based on node's communication condition:

(A) Idle
(B) Transmitting and
(C) Receiving data.

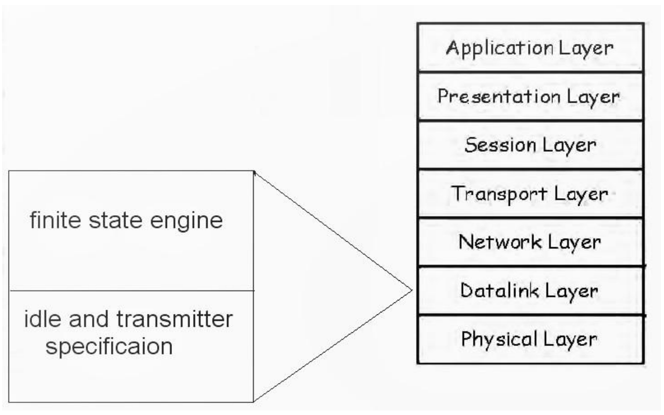


**Figure 1: Detection engine Architecture**

### A.Idle node design

This design describes the operation in an idle condition when the node is not transmitting or receiving any packet. The engine initializes at state qa and begins monitoring the host node for any new packets that are ready for incoming RTS packets or transmission. In below figure if the monitored node assembles a packet for transmission, then the engine moves to qb state otherwise, if an RTS packet is received by the node then, the engine moves to qc state.

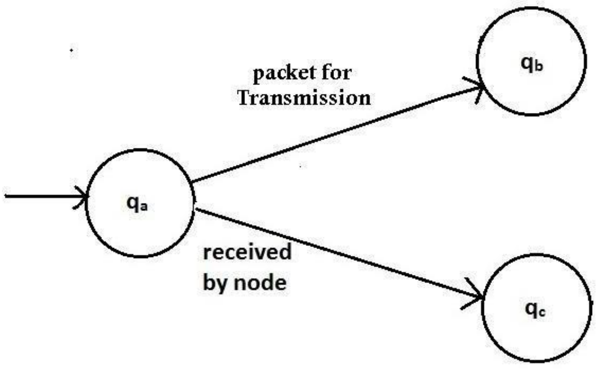


**Figure 2: Ideal node design**

### B. Transmitter design

This design explains the operation of the 802.11 MAC protocols it applies when the data is transmitted to another node in ad hoc network. When the host node assembled a new packet the engine starts at state qb, which is ready for transmission. Here, the engine checks if the communication channel is busy or idle. If the channel is idle, then the engine moves to q0 else, if the channel is busy, the engine moves to q7. In state q7 the protocol must call the back off mechanism while in q0, the expected behavior of the protocol is to transmit the RTS packet.

- **RTS design**

During the transmission of an RTS packet by the monitored node, the RTS design exemplifies the correct operation of the protocol.

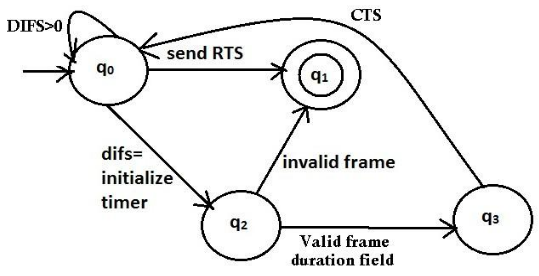


**Figure 3: RTS design.**

First, the engine retrieves the DIFS parameter from the physical layer and remains at q0 until the DIFS timer that is feeded by the DIFS parameter expires. If the node attempts to transmit an RTS before the expiration of DIFS, the engine reaches the final state q1, which designates a malicious behavior. Otherwise, it moves to q2 state. In this state, the engine checks if the frame duration field advertised by the RTS packet is the actual size of the data to be transmitted. If not, the engine moves to the final state q1, which designates a malicious behavior. Else, the monitored node transmits the RTS packet and the engine moves to q3. At this state, the monitored node has successfully transmitted an RTS and waits for CTS.

- **CTS design**

During the receiver of a CTS packet by the monitored node this design exemplifies the correct operation of the protocol. In figure 4.3, the engine monitors for incoming CTS packets. If the node does not transmit any data or attempts to transmit them before the
SIFS timer expires, then the engine moves to the final state q6, which designates a malicious behavior. Else, it moves to q5, which is the last state of the CTS design.

If a CTS packet is received prior to the CTS timer expires, the engine moves to q4. In this state, the engine retrieves the Short Inter Frame Space parameter from the physical layer and remains at this state until the SIFS timer, feed by SIFS parameter, expires. Subsequently, the engine checks for the transmission of the actual data by the monitored node.

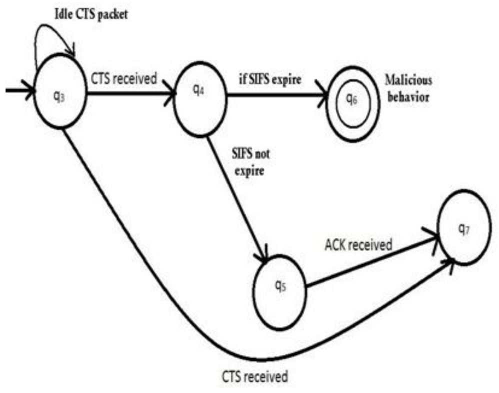
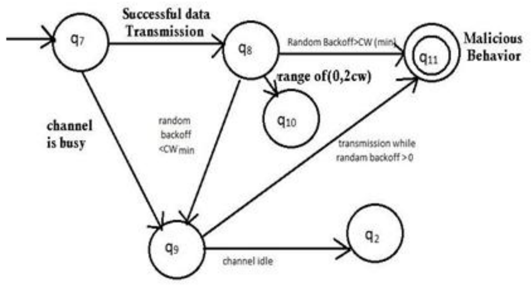


**Figure 4: CTS design**

The engine remains at this state of nodes it waits for an Acknowledgement packet, until the Acknowledgement timer expires or the Acknowledgement packet is received. Finally, the state will change at q7.

- Back off design

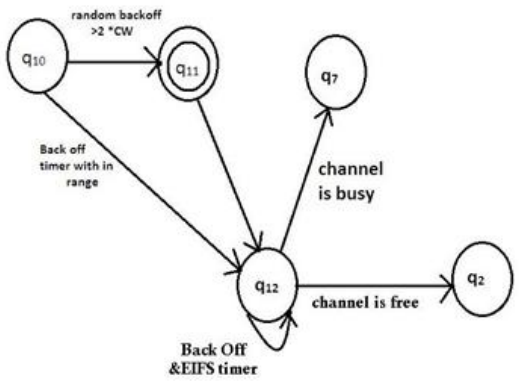


**Part (A)**

With the help of back off model the Congestion control achieved by dynamically choosing the contention window cw.

- Node increases its contention window if a node fails to receive CTS in response to its RTS.
- CW is doubled (up to an upper bound)
- When a node restores cw to CWmin after successfully completes a data transfer.

The total time is regarded as a sequence of intervals of Successful, Busy, Empty & Collision delay time.



**Part (B)**

**Figure 5: Back off design (A), Back off design (B).**

**C. Receiver Design:**

When node is trying to receive data in the form of packet from another node then engine retrieves the SIFS parameter from the physical layer and remains at qc until the SIFS timer expires. Whenever node attempts to transmit CTS before the expiration of SIFS, the engine reaches the final state q13 which designates a malicious behavior. Otherwise it moves to q14. At this state, the expected behavior of the node is to transmit a CTS packet. If the node transmits a Clear to Send, then the engine moves to q15 else it moves to the final state q13 designating a malicious behavior. Node at q15 the monitored node is waiting for the actual data packets and therefore, the engine will remain in this state until the data are received or data timeout timer expires. There are some cases are arising, that is given below.

- If the Data not received and the data timeout timer expires. Then the engine returns to the initial state qa.
- If the data timeout timer is expired as it reaches zero. The engine moves to the final state q13 marked a malicious node.
- If data are received before the data timeout timer expires. The engine moves to node state q16.

When the node attempts to transmit an ACK before the expiration of SIFS the engine reaches the final state q13 which denotes a malicious behavior, else it moves to q17. At this state of node, the expected behavior of the node is to transmit an ACK packet which shows complete transaction and the engine returns to the initial state qa else packet moves to the final state q13 designating a malicious behavior. In node state q16 the engine retrieves the SIFS parameter from the physical layer and remains at q16 until the SIFS timer expires.

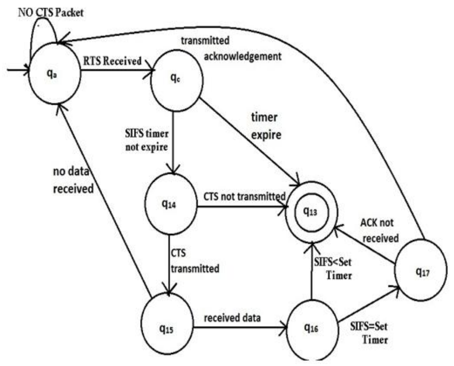


**Figure 6: Receiver design.**

Evaluation of Detection Engine

The designed engine proposed in this paper is advantageous over existing detection engines for Mobile ad hoc networks in following manner. Firstly, it resolves malicious behaviors in real time. This is important as it minimizes the time in which a malicious node can induce damage onto the network.

So Attack Detection =TD+ P + C (1)

Where TD is the time duration of frame, P is the preprocessing time, and
C is the time taken by engine to analyze the checked data.

The proposed designed engine can effectively detect all type of the attacks that target the operation of the 802.11 MAC protocol, that can be achieved by relying on operational constrains, which express any activity that does not act in accordance with these constrains will be detected and the expected protocol behavior. The concept of Signature-based detection engines monitors for predefined patterns of attacks but it’s unable to detect unknown attacks. Secondly, in above detection engine proposed when dynamic changes occur in the network or frequently changes in the topology, nodes mobility, etc. These dynamic changes can typically cause detection engines to rely on outdated information and consider decriminalise as malicious behaviors. Secondly, the proposed engine is not affected by such network conditions because node activities are monitored in real time. The proposed engine has some limitations, as detection engine cannot resolve a flooding attack or table poison attack. The detection engine at its presented state is having a tendency to false positives when hardware failures happened, in addition.

The design of the engine can be further expand in future with order to

- Detect & activate all the attacks that target the critical protocols employed at the data link layer, Network layer and transport layer of MANETs.
- Regarding hardware failures & different type of attacks, where the detection engine will be provided with a more resilient FSM design architecture in order to alleviate the need of operating a detection engine at each single network node. The design of FSM will evaluate in the contest of following steps.
- To provide better detection accuracy.
- Capability of detecting more attacks at Network & Data link layers
- False positives Rate.

Representation of MAC Layer Protocol

The IEEE 802.11 standard protocol specifies a DCF (Distributed Coordination Function) which is based on the RTS and CTS message exchange as in MACA and MACAW. In MACA and MACAW, a node in IEEE 802.11 DCF refers only until the end of CTS frame reception. It solves exposed node problem & the hidden node problem in the network. After the successful reception of the data packet only points where it differs from MACA are in the avoidance of collisions before transmitting RTS & its requirement of ACK transmission by the receiver. The scheme follows the exponential back off algorithm. Compatibly routing protocols MAC protocols are easier to manage and represent for Security. The interactions due to nature of MAC protocol, the event ordering and correct timing have crucial roles impose the necessity of using ordered models of execution with explicit timings. The explicit timing needs to be introduced in the model of event ordering due to the nature of event interactions in the MAC protocol (for timeouts explanation). The representation of IEEE 802.11 protocol in the form of Extended Finite State Machine (EFSM) has explained. There are few approaches taken in [6] and modelling of PCF protocol in [7] it is straightforward to represent 802.11 MAC layer protocol using EFSMs. The Transmissions in 802.11 MAC layer are separated by inter packet gaps known as Inter Frame Spaces (IFS). Based on different priority access Channel access permission is granted. The DIFS is used by STAs operating under the DCF for frame transmission.

A station using the DCF shall be allowed to transmit if it determines that the medium is idle after a correctly received frame, and its backoff time has expired. It has the lowest priority. SIFS is the shortest of the interframe spaces. It is used when the stations have seized the medium and need to keep it for the duration of the frame exchange sequence. The messages exchange between nodes i and j.

The introduction of TSIFS (SIFS timer), TDIFS (DIFS timer), TB (back off timer) and TOUT (timer out) has given. When a node is waiting f or a reply a timer set to a predetermined value. If the reply doesn't arrive during the specified period, timer is set into time out mode (it has expired) and the node makes transition into corresponding initial state or error state. The concept also introduces TRTS/CTS that are set to a value that is defined in RTS/CTS message when the node overhears. All timers can be either inactive or active. Normally, if its value has reached 0 the timer can be expired.

The Extended Finite State Machine (EFSM) is representation of the node that is sending data is represented. In order to send data, the node first needs to send RTS. The node makes a transition into state 1 and sets higher back off period (maximal value is

If CTS doesn't arrive during TOUT Otherwise, if CTS is received, it makes a transition into state 5, waits for TSIFS and transmits data. It waits for ACK from node j in state 6. This is represented as transition from 0 to 1. when the node waits for DIFS and backoff period to expire the transitions $1 \rightarrow 2$, $2 \rightarrow 1$, $2 \rightarrow 3$ and $3 \rightarrow 3$ represent part of the protocol that previously described When the medium becomes free and the timer is decremented to zero the node transmits RTS and makes a transition into the next state, where it waits for CTS from node j

It makes a transition to either state 0 or state 1, depending on whether the ACK j reaches node i or not. Transition from state 4 to state 1 represents the case when the destination node is either out of range or its CTS signal cannot reach the transmitter for some other reason. The case when multiple RTS signals collide and never reach the destination is also included in this transit ion since the transmitting node waits for CTS not knowing that RTS never reached the destination. In case when node i hears RTS or CTS meant for node m, where m = i, it makes a transition to state 0', where it waits for RTS/CTS and upon expiration it returns to state 0. The Finite State Machine representation of the node that is receiving data is represented.

MAC layer issues in wireless networks and Cross-layer Interaction

Routing & MAC layers interact in numerous ways. Although the authors don't address malicious behavior of nodes, it is obvious that cross-layer interaction can be abused by malicious nodes to mount DoS attack in the MAC layer and propagate it to the routing layer. The MAC layer Contention causes a routing protocol to respond by initiating new route queries Specific routes chosen by the routing protocol can significantly affect the performance of the underlying MAC protocols. This enables the intruder not only to break the existing routes, but also to maximize the probability of including himself in the new routes by maximizing the number of nodes he is disabling while minimizing the probability of being detected. In [5] the authors address the problem of selfish nodes, but the same scenario can be used by malicious nodes as well.

All communication is done at the MAC level and there are no signals that are passed to the higher levels except the final ACK signal that notifies the routing layer that the data has been successfully forwarded to the next hop. MAC layer has mechanisms to protect itself from congestions, but these mechanisms can be abused by attackers and used to disrupt communication in the MAC layer. The basic mechanism of MAC layer exchanges a series of control signals before it sends the data. If the control signals at either receiver or sender side is not received within a certain period of time, the signal is retransmitted, and it means an upper bound on the number of transmission exists. However, the failure of service at the MAC layer causes route disruption at the routing level. As we will see in this section, the attacker can use the MAC layer properties to disable and isolate several key nodes and partition the network.
Therefore, attack-resilient MAC protocol should have communication with Intrusion Detection System and routing layer both. When congestion is detected in either MAC or routing, the layer where the congestion originates should pass that information to the other layer and to the IDS. IDS should detect if the congestion is an attack and based on that decision the MAC/routing decide on future actions: to create new routes or discard the activity of the node that is causing congestion and pass that information to the other nodes. The various parameters should monitor by the system that are characteristic to MAC and routing protocols and based on their values make decisions about future actions. There are some guidelines for parameters that can be exchanged between layers are given in [3].
There are several types of attacks can be solved by the MAC layer. First of all, an attacker can keep the channel busy so that the normal node cannot use it for transmissions, which leads to DoS attack in that node. The nodes follow binary exponential backoff scheme that favors the last winner amongst the competing nodes. This leads to the capture effect where nodes that are heavily loaded tend to capture the channel by continuously transmitting data which makes lightly loaded neighbors to back off continuously. We classify a node as normal if it obeys the rules of MAC layer protocols when both sending and receiving packets based on the previous analysis. This type of nodes will not behave selfishly and will reply to RTS requests from other nodes and will update their NAV and CW, etc. according to the protocol Rule.
Finally, a node is classified as misbehaving if it denies to follow-up the rules of the protocol in order to gain priority in the network or disrupt already existing routes. This group of nodes includes wide range of behavior from malicious nodes that start misbehaving after a certain point in order to maintain the priority up to nodes that jam the network with large number of packets. Misbehaving nodes can change the value of CW, NAV value, Duration/ID field in the packet etc. A node is classified as malicious if it employs legitimate communication with other malicious or normal nodes which results in DoS in one or multiple nodes and attack propagation through the network.

**Cross-layer attacks**

In this section include both malicious and misbehaving nodes all attacks. We use the realistic scenario, where each node initially employs legal communication patterns that prevent other nodes from communicating and after some time they start misbehaving in order to maintain priority in the network.

- Attack 1

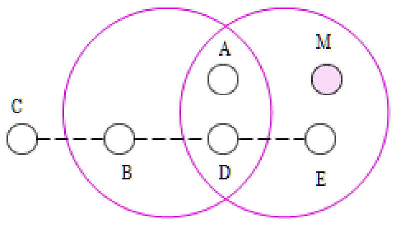


**Figure 7: Attack scenario 1**

There is such assumption that node A or D in figure 7 want to send data. The malicious node denotes as M. Node M captures the medium before node A (D) decides to send data. So, node A and D relate as back-off for TRTS/CTS. At the end of transmission node M will have to wait for tDIFS + CWmin while A will wait for tDIFS + CW, where CW > CWmin. When node A tries to send a packet it will either sense the medium busy or stay in the same loop or it will eventually collide with RTS of node M.

In this case its set of transitions is infinite loop. Node C, which wants to send a package through node D that, is in the range of node M also cannot send any data. C sends a package to B, but B cannot receive any response from D because M has captured the medium. This attack addresses the unfairness of the 802.11 protocol since node that constantly fails to send data has worse chance to be enabled to send data as time passes. Hence, it is more likely that nodes with large CW that are backing off will be declared dead by other nodes than to get an opportunity to transmit. As a moment, the throughput of the system is degraded. To be able to detect this kind of malicious behavior, cooperation of MAC and routing layers is required.

- **Attack 2**

During investigating the traffic, the attacker can find out which routes have higher priority. In the next step, mounting an attack from the MAC layer an attacker congests the channels and breaks multiple routes, increasing the possibility that in the new route search it is included in the new path. Part of the attack could increase the probability of the node being included in the new path by false route advertisements or some other method that would increase the probability of node being included in the path in case multiple paths are left in case of attack 1. The new route will be C → B → A → M if the route C → B → D → E will be broken.

- Attack 3

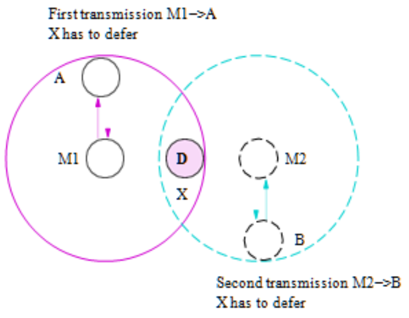


**Figure 8: Attack scenario 5.**

If user observing a system that contains two malicious nodes. These nodes are not directly cooperating because they are out of range of each other, but both are in the range of the attacked node N. The attack scenario can be performed by node that is given below. Malicious node M1 sends RTS to node A. RTS has information that the medium needs to be reserved for time t1. At time t node N receives RTS from M1 and defers its transmission for that period of time. Suppose that node M2 needs to transfer data. It sends RTS to node B TDIF1`S before the expiration of waiting period that was imposed by M1's transmission. When the first transmission stops this one starts and the medium is reserved Node M2 waits for TDIFS and exactly at time. Since M1, A and M2, B are out of reach of each other but both can be heard by node N, unless additional fairness constraints are not added they can continue their transmission infinitely many times.

Attack detection & Times based node failure recovery

We are facing one of the challenges in protecting an ad hoc network against DoS attacks, apart from distinguishing normal from abnormal traffic, is distinguishing congestions caused by malicious and non-malicious actions while minimizing the number of false alarms. We have seen in previous sections; it is not meaningful to speak about neither MAC nor routing protocol in isolation. MAC layer protocols significantly influence routing protocols and vice versa. When the data is already sent however, we have already mentioned that current interaction between MAC and routing protocols is limited to the exchange of ACK signals.

In order to mitigate the effects of congestion we need to design new dynamically adaptive protocols that can adapt to changing network and traffic characteristics by measuring and exchanging parameters that characterize cross-layer interaction and providing alternate routes with less traffic. However, in case of attacks that start in either the MAC or routing layer, providing alternate routes may represent an opportunity for the attacker to include himself in the new routes. In case when IDS relies only on measuring traffic rates the number of false alarms rapidly increases. Hence, when incorporating cross-layer interaction we need to include interaction with an Intrusion Detection System. This implies the necessity of introducing a more complex system that would observe both traffic rates and several other protocol-related parameters, such as NAV, CW injection rate, etc. and impose timing constraints.

The routing layers and MAC would have to cooperate with each other in order to avoid points of congestion and reroute traffic and with the IDS in order to isolate malicious nodes or to avoid inclusion of malicious nodes in the new routes and propagate the information throughout the network. For attack detection we use the proposed EFSM models of communicating nodes presented in III. Additional parameters are needed for determining the nature of transitions in order to avoid false alarms. Due to the fact that nodes in ad hoc networks cannot be perfectly synchronized due to mobility and other parameters, the nodes participating in attacks presented in IV start misbehaving after some time due to the fact that absolute synchronization that leads to blocking of targeted nodes cannot be maintained in real ad hoc network. In order to perform the attacker(s) need to violate one or more rules & the attack without letting the attacked node(s) communicate. As it can be seen from figure there are several possible cases that lead to routes break.

A node may not be able to receive any requests from neighboring nodes due to a high traffic rate. If it is already included in a route when the congestion starts, it will not be able to respond to any requests and eventually the connection times out and the route is broken. In case when the observed node is attacked, the attacker will have to change some parameters, i.e. CW size or the value of NAV in order to gain priority and stop the node from sending packets. In this case the IDS should notify routing & MAC layers that there are no malicious nodes and that they are free to include any node in the new route. The connection times out and the new RREQ is sent.

While minimizing the probability of detection the malicious node will maximize the probability of including itself in the new route by blocking as many nodes in its vicinity as possible. In this case IDS should detect node misbehavior by controlling critical parameters that are exchanged in communication and marks node M as malicious. We can see, as observing loops in the EFSM model provides information about possible sources of attacks, but in order to distinguish between attack and congestion additional parameters or timing constraints are needed. So, we need to come up with the unified automatic approach for detection of wide range of attacks on wireless MAC protocols.

In addition, there exists a need for creating a database of attacks that cover a significant range of attacks and is used as input for IDS. For attack detection we formulate theorems that represent series of rules a fault-free MAC protocol cannot violate. Each property is formalized as a logical formula using temporal logic. We propose using Computational Tree Logic (CTL) for attack detection Automatic Model Checking is executed with input of the relevant rule (theorem) parameters from the nodes under examination. There is some general task is to check for a given CTL model M, state $s \in S$ and CTL formula $\varphi$ whether the property $\varphi$ is valid in state s of model M: $M, s \models \varphi$. For example, we define the rule that prohibits two processes to be in their critical section at the same time as:

$$AG\,(\neg\,(Pi.\,s = C \wedge Pj.\,s = C))$$

Where Pi. s, Pj. C represents a critical section and s represents states of a process. An important rule that excludes the appearance of infinite loops says that a process that wants to enter its critical section is eventually able to do so and is represented in CTL as,

AG (Pi. s = A ⇒ AF (Pi. s = C))

Where A stand for an attempt. When the negated CTL formula is accepted by the EFSM in case, the safety of the system is endangered. The EFSM execution path is used for automatic attack generation. The model checker explores the search space for errors and generates a set of error scenarios that can later be used of protocol testing. We choose a specific set of parameters and in the case of an attack we save the parameters that differ from the normal values and add the specified set of parameters to the specific states in previously generated error scenarios. For that purpose, we need to identify parameters that are used for error or attack detection. A useful extension is addition of invariant constraints that must hold in every reachable state of the observed mode. We present the results of proposed attacks on IEEE 802.11 MAC.

The experimental results do not incorporate any elements of cross-layer cooperation for now. For better illustration of the above attacks, we have used QUALNET to simulate the behavior of nodes under the attack. We simulate network traffic with duration of 120 seconds. In the first half of the simulation, the malicious node is inactive and node D is able to send packets to its neighbor. After 70s, the malicious node starts attacking node D. The second scenario is presented in 4. It is important to realize that the malicious node does not need to send the traffic to node D in order to disrupt its traffic.

The main intention of node M is to broadcast the RTS packet to node D, so it updates its Network Allocation Vector and doesn’t send any traffic for the communication period indicated in the duration field of the RTS packet.

This attack scenario requires synchronization between two malicious nodes M1 and M2. The nodes need to alternate while sending traffic and therefore they need to generate packets at half the rate of the previous scenario. The major disadvantage of this attack is that it is more difficult to detect the attacked node for data traffic generation rates of the malicious nodes. In the above figure node D is still able to send its packets. When the attack is mounted, node D is completely disrupted during the 30s attacking period.

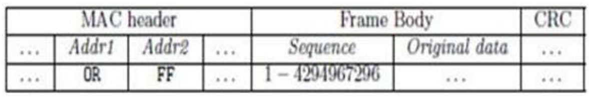
The inter-actor connectivity is a very crucial issue to maintain network operation in the wireless sensor and actor networks. Most of the applications have been proposed for harsh environments where the backbone actor nodes are prone to failure or get damaged due to their battery power exhaustion or get physically damaged. Such failures can partition the network due to failure of the cut-vertex node and eventually decrease the network performance or even sometimes make the network useless. Currently, a few approaches have been proposed to restore the partitioned network due to failure of the cut-vertex node but without considering the recovery node capabilities.The chapter proposes a localized hybrid timer based cut-vertex node failure recovery approach called distributed prioritized connectivity restoration algorithm (DPCRA) to handle such partitions and restore connectivity with the help of a small number of nodes. The main idea is to proactively identify whether the failure of an actor node causes partition or not in the network. If partition occurs the designated failure handlers (FHs) detect that partition and repair it locally using minimum information stored in each actor node. In case first designated node is unable to start the recovery process within a permissible reaction time the next designated FH could start the recovery process [5]. The main strength of chapter is the use of multiple backup nodes for the guaranteed partitioned recovery.

Header format SSF

The SSF protocol simply adds a sequence number to a normal Ethernet frame. The frame is shown in Table 1.

OR……OR: OR: OR: OR: OR is the source MAC address. FF……. FF: FF: FF: FF: FF: FF is the MAC broadcast address. Sequence….is the SSF sequence number.

Original Data…The original payload for the Ethernet frame.



| MAC header | | | | Frame Body | | CRC |
|---|---|---|---|---|---|---|
| … | *Addr1* | *Addr2* | … | *Sequence* | *Original data* | … |
| … | 0R | FF | … | 1 – 4294967296 | … | … |

**Table 1: Header format SSF**

This implementation is somewhat theoretical, and a real world implementation would want to change the protocol type in the Ethernet frame and store the old one in the SSF header. This would allow compatibility with already existing protocols. For this work, however, our solution suffices.

Node failure Prevention Using NBAC (Non-blocking Atomic Commitment) Protocol for Synchronous Systems:

During our studies, we noticed that the complexity of the group checking and group proposals, which require the use of a non-blocking atomic commitment protocol, is very high. The complexity is about N+2• N2 for an unreliable network with no node failures when a reliable multicast service using an ACK based protocol is used. The first term is for the initial message together with the required ACKs and the last term is for the votes and the finalization. In a network where node failures can happen the message complexity increases even more because a consensus algorithm is needed. The actual performance of these algorithms can be improved, if some of the networking assumptions are weakened.

Therefore, we decided to think about some possible alternate implementations of these components, and we would like to show our results in this chapter. We started with the classic paper from [5] which presents approaches for solving the non-blocking atomic commitment problem in synchronous and asynchronous networks. Because our primary application target is wireless ad hoc networks, we looked at typical properties of wireless ad hoc networks and came up with the following list.

• If a wireless frame is transmitted over a wireless medium and there are no intermediate nodes, i.e., no additional routing or forwarding of messages is performed, and then the transmission time can be easily bounded if some additional assumptions are taken. Basically, one has to ensure that these packets are not queued at either the sender- or receiver interface and are processed as fast as possible. Using appropriate scheduling techniques real-time processing within a bounded time is possible.

• If such a bounded time for transmission can be established, wireless messages are either received within a given time frame d or not at all, i.e., they are lost. Furthermore, the fact that a message has been lost can be detected by the receiver, assuming that the receiver knows that a message was sent.

• A wireless medium is a broadcast channel and therefore all algorithms should make use of this fact to reduce message complexity. This is particularly important for tasks like consensus, the exchange of votes, where we have a set of nodes and all nodes need to know the same information from a single node. These properties are mentioned here only as an introduction, and a more formal list is provided later in this chapter. However, looking at these properties we can see that this allows the implementation of a synchronous system model with unreliable links. Therefore we have chosen a basic NBAC protocol for a synchronous network.The differences are that the appropriate algorithm requires the use of an NBAC protocol with the modifications that we can piggy-back additional information with the votes, run multiple instances in parallel, and we have finalization phase.

Some of the data types used are not obvious and therefore require an additional explanation. The mapping vote is used to collect the votes from all participants for a given NBAC UID (unique identifier). A NBAC UID is a unique global id which is implemented by a local sequence counter and the unique MAC address of a node. The functions ’vote IMPL’ and ’decision IMPL’ are instances of an application dependent decision and voting function

Listing 1: A NBAC protocol for synchronous systems:

the algorithm in Listing 1 solves the NBAC problem if all networking primitives are available. The properties and the definition of the NBAC problem, as well as the basic networking primitives, are repeated below for presentation purposes and are based on [2].

A NBAC protocol assures that all participants take the same decision, which is either COMMIT or ABORT. The exact meaning is application dependent, but committing
typically it means making changes permanent and abort cancels all work. Whether the outcome is COMMIT or ABORT depends on the votes from the individual participants and on network or node failures.

Non-blocking atomic commitment - NBAC.

A protocol solving the NBAC problem satisfies the following properties:

Termination: Every correct participant eventually decides.

Integrity: A participant decides at most once.

Uniform agreement: No two participants decide differently.

Validity: If a participant decides COMMIT, then all participants have voted COMMIT.

Non-Triviality: If all participants vote COMMIT and there is no failure the outcome decision is COMMIT. Let P = {p} be a set of participants. The most basic primitive is multicasts end, which sends a message containing some information to a set of participants. If multicast addresses are available for every possible group, an implementation would simply send the message to this address. In a broadcast network an efficient implementation is to send the message to the broadcast address and include the set of participants in the data payload. This data payload is then inspected by the receiving nodes and if the message is targeted for that node it is delivered at node. An obvious drawback of this primitive is that it is not fault-tolerant because it cannot be guaranteed that all nodes of P will actually deliver the message.

A very powerful primitive is the s_rel_multicast send which reliably sends a message m to a set of process with reliable multicast in synchronous systems. The aim of the primitive S_rel_multicast send (P, m) is to reliably send a message m to all participant's P with an all-or-none atomicity property.

Termination: If a correct process p multicast a message m to the set of participants P, then some correct process delivers m (or all processes are faulty).

Validity: If a process p delivers a message m, then m has been multicast to a set of participant's P and p belongs to this set.

Integrity: A process p delivers a message m at most once.

Uniform agreement: If any process of P delivers m, then all correct processes deliver m.

Timeliness: There is a time constant d such that if the multicast is initiated at time t, no process delivers a message m after t + d'.

The big problem is that it is impossible to implement the synchronous reliable multi- cast primitive in a network with unreliable links.

We will start by showing how the reliable multicast primitive can be implemented in our modified network model and will then analyze the complexity of the complete algorithm, which was the primary reason for seeking an alternate approach. In the second part of this section, we will present an alternate solution which solves the same problem using a different approach but with a better message complexity.

Implementation using the Synchronous Reliable Multicast

We have already proved in Theorem 2 that in general it is not possible to implement a reliable multicast if the links are unreliable and an unbounded number of links can fail. But this situation is different in our modified network model. We start by claiming the following Lemma.

Lemma 1. Let P = {p} be a set of participants which are fully connected and assume our system model from Definition assume that process P1. . . Pn wants to multicast a message to all participants. If every node broadcasts the message again before locally delivering it we can guarantee for f = n - 2 link and node failures in total, that either all correct nodes deliver the multicast message m within two rounds (equaling 2d), or it is not delivered at all.

Proof. The first case is that no participants deliver the message. Let us assume that p crashes before multicasting the message to all participants. Then no participant including p1 delivers the message. The reason for this is that p by itself does only deliver the message locally after it has multicasted it to all participants. Furthermore, the message has never been sent over the multicast channel and therefore no other node can receive it. Now assume that p1manages to broadcast its value. In that case, the receiver set gets parted into two sets, P respectively P of nodes that did, respectively did not receive the message from p1

Clearly |P2| = f'2= f, and |P1| = n -1 -f'. In the second round, all correct processes in P1attempt to forward the message from p. As there are only
e = f - f' errors left, with f = n - 2, at least n - 1 - f'1- e = n - 1 - f = 1 processes in P1 will not be hit by faults in the second round and thus forward p’s message to all processes in P2.

Listing 2 shows an algorithm which solves this problem in our modified network model and we claim the following properties:

Theorem 3. The algorithm shown in Listing 2 implements the reliable multicast primitive defined. A message can be sent by invoking s_rel_multicast send and any message delivered is passed to the application by the Invocation of s_rel _multicast deliver.

Proof. Let P = {p} be a set of multicast participants and let us assume w.l.o.g that p1. . . Pn is the process which has initiated the reliable multicast by invoking the primitive s rel multicast send. We will first deal with the special case when p crashes before executing line 23. Then no process receives the message and all properties hold trivially. Now let us assume that p executes line 23. The algorithm is essentially an implementation of the algorithm described in Lemma 1 and therefore we have the Uniform Agreement property. Validity is obvious from the algorithm because the set of participants is included in the message and it is checked in Line 48 and only values sent from participants belonging to this set are taken into account. Termination is also simple. If the process is correct it can send the message in line 23.

If there is existence another correct process it receives m (see the beginning of this proof). Because it has not received the message before, the test in line 54 will pass and it will deliver the message giving us the desired result. Integrity is also implemented by the check in Line 54. If a message has been received its sequence number is added to this set. Sequence numbers are only removed after 2d which is essentially the time required for our broadcasting algorithm as shown in Lemma 1. Therefore, it is safe to remove the old sequence numbers afterwards

Theorem 4. In our network model, the message complexity of the algorithm shown in Listing 4.2 is in T (n).

Proof. There are n participants. At the beginning the initiator sends the message, which accounts for 1 message. At most n - 1 participant can receive the message. Every participant only forwards the message once, and therefore we have n messages in total. Therefore, the message complexity is constant giving us the desired result.

Theorem 5. There exists a time constant d'= 2d for the algorithm shown in Listing 4.2 such that if the multicast is initiated at time t, then no process delivers the multicast after the time t + d

Proof. If the initiator of the multicast sends a message at time t, then all processes receive this message at most at t + d by our network model assumption. If the local processing takes no time, which we assume here, then every correct process which has receive the message broadcasts it again. Again these messages take at most d. Furthermore, a process only forwards a message once, which gives our desired result.

Listing 3: Simple and reliable multicast networking primitives

Practical implementation concerns

If one wants to implement the algorithms shown in Listing 2 the following things should be considered.

- The variable ID is simply the MAC address of the first network interface of this node. This provides a unique global address assuming that the network configuration is correct.
- Assuming a wireless network with an IEEE802.11 network layer the function low- level multicast should be implemented by using the mgs type as the Ethernet frame type and by sending to the MAC broadcast address.
- The function low-level receive should take all messages received by the MAC layer. If the

message is either of the type MSG TYPE SMCAST or MSG TYPE RMCAST is should be processed by the appropriate functions. Otherwise, the default handler should be used.

• If node recovery is required the local sequence counter must be updated in a safe manner, for example, by using a local transaction spanning the entire multicast. Furthermore, the remote sequence counters have to be restored on start up.

Implementation of the NBAC (Non-blocking atomic commitment)

Having proved that such a reliable multicast primitive can be implemented we can use the synchronous NBAC algorithm from Raynal to implement the author algorithm. This is shown in Listing 3. The correctness of the NBAC is shown in [2].

The finalization concept is easy in a synchronous system because it simple has to ensure that every participant has executed the decision.

Definition 35 (Finalization) A protocol solving NBAC finalization in the author sense has the following properties:

Local Finalization: Eventually finalize IMPL will be called on each correct participant after decision IMPL has terminated on this participant.

Finalization Agreement: No two participants decide on different finalize results. Finalization Validity: If a participant decides on the finalize result COMMIT, then decision has terminated on every participant.

Finalization Non-Triviality: If there is no failure suspicion, then the finalize result is COMMIT.

Theorem 6. The algorithm shown in Listing 3 solves the finalization problem.

Proof. Finalization agreement is simple because we use the same result value as for the NBAC. Since the NBAC guarantees uniform agreement we get the finalization agreement for free. The finalization validity property is easy to achieve in a synchronous system because the execution of statements can differ by at most the time d at any node if we can neglect computation times. The reason for this is that the only time which depends on the network is the reception of the initial multicast message. Therefore, the property can easily be implemented by waiting the time d in line 49. Finalization non-triviality is also simple. If all nodes have sent their votes and their votes are COMMIT, then the result is COMMIT and so is the finalize result. Local finalization is simple because the code is sequential and the execution of finalization IMPL is after decision IMPL.

Listing 4: NBAC protocol for the TBUT network model Code attached in Annexure; Listing 3

Theorem 7. The total message complexity of the NBAC for n participants is in O (n2).

Proof. Initially the initiator initiates the NBAC by sending a simple multicast message to all participants. Using our multicast implementation this accounts for 1 message. Every node which participates in the multicast must multicast its vote to all other nodes using a reliable multicast. One reliable multicast requires n messages and therefore n reliable multicasts require n2 message. In total we have n.

**Implementation using Agreement**

A message which is O (n) in our second algorithm only uses a simple multicast and uses an additional agreement phase. Using our network model, it allows a very efficient implementation which has a message complexity of O (n). We start by replacing the srel multicast send primitive with the normal multicast primitive. The resulting algorithm is shown in Listing 4.

Warning - The algorithm below should NOT be used for anything this shown here for explanation purposes.

Listing 5: Bad NBAC protocol for the TBUT network model

Listing 6:An example which violates the uniform agreement is easily given. Assume that there are 4 nodes, where Figure 12 shows the transmission graph of the network.

Node 1 initiates the NBAC protocol and sends a message to the participants 1, 2, 3 and 4.

Node 1 receives the message and votes COMMIT and broadcast its vote.

Node 2, 3, and 4 also receive the message and vote COMMIT.

Node 1 would receive all votes and therefore would vote COMMIT. The same holds for node 3.

Node 2 would miss the vote from node 4 and node 4 the vote from node 2. Therefore, these votes would be ABORT.

We now have nodes 1 and 3 voting COMMIT and nodes 2 and 4 voting ABORT. This violates the uniform agreement property. Therefore, we have chosen to add an additional and optimized agreement protocol which assures that the variable result is consistent among all nodes. The basic structure is shown in Listing 5. The algorithm chosen for our agreement, which is shown in Listing 4.6, can tolerate f failures, where f = n -2, and requires 2 synchronous rounds.

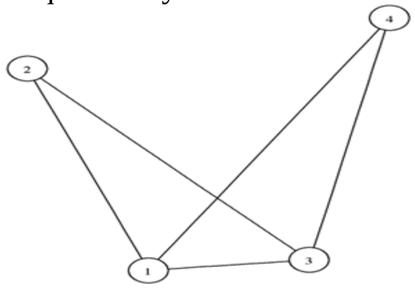


**Figure 9: Counter example for modified NBAC.**

In each round, every node broadcasts its current set of votes to all nodes. After 2 rounds, the nodes choose the minimum value from their current set, which is taken as the input to the decision. We will show the complete algorithm in the following section together with its correctness proof.

Listing 5:Agreement protocol

We will now show how this can be achieved by using our agreement protocol. The variable V holds the exchanged votes. The variable queue is a global variable and holds the round k messages for a given NBAC instance. The variable round holds the current active round for a given NBAC.

Listing 6:  Agreement protocol in the TBUT network model.

First we note that all nodes will start at most d apart if 'agreement' algorithm is executed from within Listing 6 The reason is that the only time which depends on the network is the initial reception of the multicast message from the originator. Therefore, the execution is aligned on grids. Note that there are always overlapping areas for every round. The carefully reader will notice that enforcing this grid actually requires our network model.

The local clocks are not allowed to drift unbounded to each other within a given time frame d. This fact has been taken into account by the maximum clock jitter J. The time required to wait increases with every new round because of this jitter.

## Conclusion

In this chapter, we have developed a finite state machine to evaluate the performance of packets in different situation such as (A) Idle (B) transmitting and (C) receiving data. In order to simplify the presentation of the detection engine with detailed description of design model describe engine is work as system model on NIC and Provide secure channel when node communicate to each other.

Finally in network where node failures can happen due to more packet or packet queue broadcast in the network (message complexity increases) even more because a consensus algorithm is needed. The actual performance of these algorithms can be improved, if some of the networking assumptions are weakened. Therefore, we decided to think about some possible alternate implementations of these components, using NBAC algorithm.

The actual performance of these algorithms can be improved, if some of the networking assumptions are weakened. Therefore, we have presents approaches for solving the non-blocking atomic commitment problem in synchronous and asynchronous networks. A NBAC protocol assures that all participants take the same decision, which is either COMMIT or ABORT. The exact meaning is application dependent, but committing typically it means making changes permanent and abort cancels all work. Whether the outcome is COMMIT or ABORT depends on the votes from the individual participants and on network or node failures. So the approaches are discussed for resolve problem of packet collision on broadcast in situation of queue with focus of identify malicious node.

## References:

[1]. Majumder, S., & Bhattacharyya, D. (2018, January). "Mitigating wormhole attack in MANET using absolute deviation

statistical approach" In 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 317-320). IEEE.

[2]. Ali, A. K., Sharma, B., & Sharma, U. M. (2016). "Impact Analysis [12]. Rajakumar, P., Prasanna, V. T., & Pitchaikkannu, A. (2014, February).

[3]. "Security attacks and detection schemes in MANET" In 2014 International Conference on Electronics and Communication Systems (ICECS) (pp. 1-6). IEEE

[4]. Kumar, A., & Singh, J. (2015). "Security Attacks in Mobile Ad hoc Networks (MANET): A Literature Survey" International Journal of Computer Applications, Vol. 122 Issue 20, pp. 31-35, July 2015.

[5]. B. Vanathy and M. Ramakrishnan, "Signcryption based hyper elliptical cureve cryptography framework for key escrow in manet," International Journal of Advanced Research in Engineering and Technology (IJARET), vol. 11, no. 3, pp. 91–107, 2020

[6]. H. Shan, H. Cheng, and W. Zhuang, Aug. 2011 "Cross-Layer Cooperative MAC Protocol in Distributed Wireless Networks," *IEEE Trans. Wireless Communication*, vol.10, no.8, pp.2603-2615.

[7]. Jain Sachin "Reliability Assessment and Feasibility Study of Software Metrics in Object Oriented Environment" ISSN: 2454-4248 Volume: 5 Issue: 5 pp- 369-373

[8]. Li Shi-Chang, Yang Hao-Lan and Zhu Qing-Sheng, June 2010 "Research on MANET Security Architecture Design", In IEEE Conference on Signal Acquisition and processing, pp. 90-93.

[9]. H. Deng, W. Li, D. P. Agrawal, *Routing Security in Wireless Ad Hoc Networks*, IEEE Comm. Magazine, October 2002.

[10]. S. Marti, T. J. Giuli, K. Lai and M. Baker, *Mitigat- ing Routing Misbehavior in Mobile Ad Hoc Networks*,

[11]. MOBICOM 2000, Boston, MA

[12]. C. Barrett, M. Drozda, A. Marathe, M. V. Marathe *Analyzing Interaction between network protocols, topology and traffic in wireless radio networks*, Proc. IEEE Wireless Comm. and Networking Conference (WCNC'03), New Orleans, Louisiana, 2003.

[13]. V. Gupta, S. Krishnamurthy, M. Faloutsos *Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks*, IEEE Milcom 2002, Anaheim, California, October 7-10,

[14]. Jain Sachin, "Improving the Performance of Heterogeneous Hadoop Clusters Using the Map Reduce Big Data Algorithm", International Journal of Advanced Science and Technology.