Formal Verification of a Cooperative Automatic Repeat Request MAC Protocol

Xin He, Ram Kumar, Liping Mu, Terje Gjøsæter and Frank Y. Li

Dept. of Information and Communication Technology, University of Agder (UiA), N-4898 Grimstad, Norway
{xin.he, liping.mu, ram.kumar, terje.gjosater, frank.li}@uia.no

Abstract

Cooperative communications, in which a relay node helps the source node to deliver its packets to the destination node, are able to obtain significant benefits in terms of transmission reliability, coverage extension and energy efficiency. A Cooperative Automatic Repeat reQuest (*C-ARQ*) MAC protocol has been recently proposed to exploit cooperative diversity at the MAC layer. In this paper, we validate the integrity and the validity of the *C-ARQ* protocol using formal methods. The protocol logic is modeled in SDL and implemented in PROMELA. The functionality of the C-ARQ protocol is verified through simulations and verifications using SPIN.

Keywords: Cooperative communications, Finite model-checking, Protocol verification, PROMELA

1. Introduction

With the development of advanced radio communication technologies, wireless networks have been widely accepted as a last-mile solution to provide ubiquitous access to Internet services. Different from wired transmission, broadcast is an inherent feature of wireless transmission, i.e., the information transmitted from a source node can be received not only by the destination node, but also by neighboring nodes surrounding the source. In traditional wireless networks, such signals received by the neighboring nodes are treated as interference and many techniques have been developed to alleviate their impact. However, such signals actually contain useful information for the destination node. In fact, if the information can be properly forwarded by the surrounding node(s), the reception performance at the destination could be improved. This fact indeed motivates the application of a new technology, known as cooperative communications [1].

The theory behind cooperative communications has been studied in depth and significant improvement of system performance has been demonstrated in terms of transmission reliability, network coverage and energy efficiency [2]. Meanwhile, cooperative Media Access Control (MAC) protocol design for wireless distributed networks is attracting more and more attention within the research community [3, 4, 5]. From the MAC design perspective, three key issues need to be addressed. a) *when to cooperate*: Since the quality of wireless channels varies with time, a source node may not always need help from relay nodes. Therefore, it is more sensible that cooperation is initiated only when it is necessary and beneficial. b) *whom to cooperate with*: One or more appropriate relay nodes need to be selected among multiple potential relay nodes in the network. In a distributed network where there is no central controller that can coordinate data transmissions of all relays, relay selection becomes a challenging task. Without an efficient relay selection scheme, collisions might happen frequently when several potential relays are contending for channel access at the same time. c) *how to protect cooperative transmissions*: The MAC protocol should be carefully designed to protect all ongoing transmission sequences against potential collisions from any other nodes in the vicinity.

A novel Cooperative Automatic Repeat reQuest protocol (C-ARQ) which addresses the aforementioned three issues concerning cooperative transmissions at the MAC layer has been proposed [6] based on the Distributed Coordination Function (DCF) scheme for Wireless Local Networks. According to C-ARQ, cooperative transmission is initiated only when the direct transmission fails. In this way, unnecessary occupation of channels by relay nodes and waste of system resources are avoided. Secondly, the relay nodes are sorted with different backoff time before data transmission according to instantaneous relay channel quality, and the relay node with best relay channel quality will be selected automatically to forward the data packet first. Finally, the cooperative transmission sequences are specifically designed to give cooperative retransmissions higher priority for channel access in order to protect ongoing packet forwarding by relay nodes.

Numerous approaches exist to verify the correctness and feasibility of a new protocol. One is by experimentation in real-life scenarios. Another well-known approach is via the use of formal methods. Model checking is one such method, which consists of constructing a computer tractable description (formal model) of the protocol and then using a specific automatic (or semi-automatic) analysis technique to prove or to check the satisfaction of a given set of critical properties [7, 8]. Model checking can be used to formally verify finite-state concurrent systems. Specifications about the system are expressed as temporal logic formulas. Symbolic algorithms are used to traverse the model defined by the system and check if the specification holds or not. Analyzing a protocol with model checking

consists of primarily constructing an abstract description, or a model, of the protocol with the main features that could produce execution errors. The reliability properties of the protocol are specified using a property-oriented language. Finally, the reachability graph is produced including all the execution paths for the model in order to check whether these paths satisfy the properties. It has been widely used to formally verify communication protocols with model checking techniques. Related work can be found in references [10]−[14].

The goal of this paper is to verify the functionality of the C-ARQ MAC protocol using formal model checking methods. To do so, we use the *Specification and Description Language* (SDL) to specify and visualize a formal model for the cooperative system. Furthermore, *Process or Protocol Meta Language* (PROMELA) is employed along with the Sequential Programming in PROMELA (SPIN) model checker to verify the integrity and validity of the C-ARQ protocol. Different network scenarios are simulated, and verifications are carried out through never-claims, using the Linear Temporal Logic (LTL) formula in SPIN.

The remainder of the paper is organized as follows. In Section 2, the system model and the C-ARQ protocol are explained in details. In Section 3, we introduce the SPIN model checker and PROMELA. In further sections, we describe the SDL model for our protocol (Section 4), and explain the simulation and verification results (Sections 5 and 6) from the SPIN model checker. The main challenges and difficulties of the model checking of the proposal is summarized in (Section 7). Finally, we conclude the paper in Section. 8.

2. C-ARQ Protocol Description

2.1. System Model for Cooperative Transmission

The C-ARQ protocol is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism in a single-hop scenario, where the source and the destination can hear each other, and there is only one node sending packets at any time. Hence, we use the network in Fig.1 for illustrating the cooperative protocol procedure [6]. The network consists of a source node, S, a destination node, D, and several other randomly distributed potential relay nodes, R_1, R_2, ... R_n. In

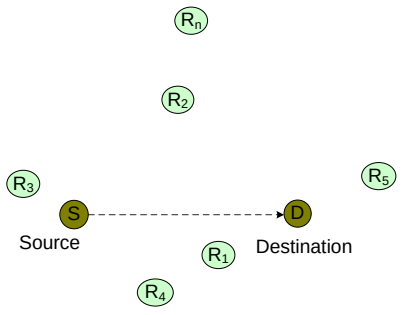


Figure 1: System Model for Cooperative Transmission.

this model, S and D are within the transmission range of each other. Each packet transmission cycle starts from S, with the intended destination as D. Other nodes in the network that can hear both the source node and the destination node and have correctly decoded the data packets they capture from the direct link become relay candidates. The cooperative retransmission is initiated only when the direct transmission fails. The relay nodes with top relay channel quality will be automatically selected to forward the DATA packet to the destination following the C-ARQ protocol.

Furthermore, this network model can be easily extended to a multi-hop scenario, where the single-hop link in our model, including the relay nodes, acts as a virtual node along the whole transmission path.

2.2. C-ARQ MAC Protocol Description

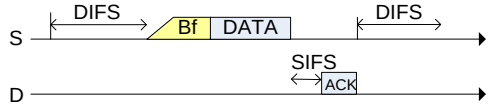
The message exchange sequence of the C-ARQ basic access scheme is illustrated in Fig.2 [6]. It has four operation cases, as Case I: direct transmission succeeds; Case II: best-relay-channel retransmission succeeds; Case III: multi-relay retransmission succeeds; and Case IV: the whole cooperative retransmission fails. These cases are further elaborated in the following.

Case I: As the first step, the source node, S, sends out a DATA packet to destination D following the original DCF basic access scheme [9]. According to the DCF protocol shown in Fig.2(a), S listens to the channel for a DCF InterFrame Space (DIFS) interval and then waits for a random backoff time before transmission in order to avoid possible collision. If the transmission succeeds, an acknowledgment (ACK) frame will be returned to the source node after a Short InterFrame Space (SIFS) interval.

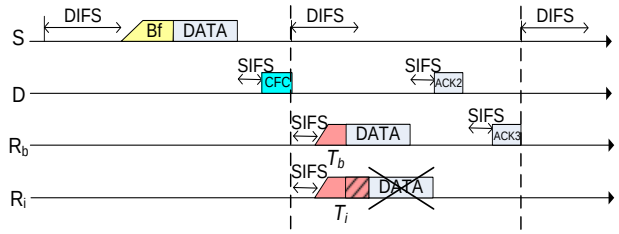
Case II: As mentioned earlier, if and only if the data packet is received erroneously at D, the cooperative phase will be initiated. The error-check can be performed by means of Cyclic Redundancy Check (CRC). As shown in Fig.2(b), D broadcasts a Call For Cooperation (CFC) packet after SIFS to invite other nodes in the network to operate as relay nodes and at the same time provides them with the opportunity of measuring their respective relay channel quality. The CFC frame adopts a similar format as the ACK frame but with a broadcast address in its address field. It is transmitted at the basic data rate in order to invite as many relay nodes as possible. Having received both the CFC packet and the DATA packet correctly, each relay candidate will measure the signal strength of its received CFC packet, and if the Signal-to-Noise Ratio (SNR) exceeds SNR_{low}, the relay candidate will start its timer according to Eq. (1) [6].

$$T_i = \left\lfloor \frac{SNR_{low}}{SNR_i} \frac{DIFS - SIFS}{slottime} \right\rfloor, \quad i = 1, 2 \cdots n \qquad (1)$$

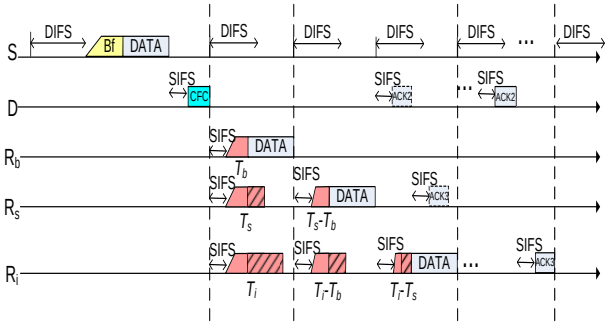
where T_i is the backoff time at relay node R_i, defined as an integer in number of microseconds; SNR_i is the SNR value in dB of the received packet from D measured at R_i; SNR_{low} is the threshold of SNR_i for R_i to participate in cooperative retransmission; and n is the number of relay nodes in the network. The upper bound of the backoff time for relay candidates is $DIFS - SIFS$ in order to prioritize the relay nodes for cooperative retransmissions among other contending nodes in the



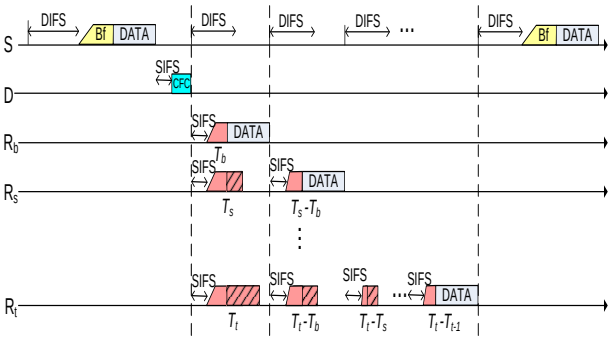
(a) Case I: Direct Transmission Succeeds.



(b) Case II: Best-relay-channel Retransmission Succeeds.



(c) Case III: Multi-relay Cooperation Succeeds.



(d) Case IV: Cooperative Retransmission Fails.

Figure 2: C-ARQ Basic Access Scheme.

network. The granularity of T_i is specified to be *slottime* of the system in order to cover the propagation delay in the network.

According to Eq.(1), the relay node with the best relay channel quality, R_b, which observes the strongest received signal and thus has the shortest backoff time T_b, will first get access to the channel and forward its received packet to the destination. When the other relay candidates hear the packet sent by R_b, they will freeze their timer and keep on listening to the channel. More details about the automatic relay selection scheme can be found in [6].

Moreover, the granularity of the back-off time, T_i, can be adjusted according to specific network scenarios. For example, the granularity can be refined to 1 us, which is enough to distinguish signals sent from two stations that are 300 meters away from each other. In our description, we use slottime as the granularity in order to keep the legacy of the 802.11 standard. In this case, there is only two slottime between DIFS and SIFS, which means three backoff duration, can be allocated to different relays, i.e. 0, 1 or 2 slottime. This scheme works well on a sparse network with less than 10 relay nodes [15]. However, the performance will be degraded due to high collision probabilities among relays in a dense network. Hence, an optimization scheme has been proposed in another follow-up work [16], aiming at minimizing the relay collision probability in a dense network by optimizing the backoff time allocation scheme. Significant throughput enhancement has been shown by both analysis and simulations when the proposed optimal relay scheme is applied.

As shown in Fig.2(b), after the direct transmission fails, S keeps listening to the channel for the next data transmission. If there are no relay nodes in the network that satisfy the relay selection criterion, S will obtain the channel access after DIFS and a random backoff time. If D decodes the packet correctly after the best-relay-channel retransmission, D will return an ACK2 packet, which is relayed afterwards as ACK3 by R_b to S in order to guarantee a reliable ACK transmission. All the relay nodes will reset their timer and discard the packets they have received after they detect the ACK2 packet sent by D. Thus, the cooperative retransmission phase is completed.

Case III: Cooperative retransmission failure can be caused either by collisions among two or more relay nodes with the same backoff time T_b or by data corruption on the transmission channel. In this case, no ACK2 packet is sensed from the channel, and the other relay nodes will reactivate their timers simultaneously after the ACK timeout, as shown in Fig. 2(c). Following the same procedure as the best-relay-channel retransmission, the timer of the relay node with the second-best-relay-channel R_s will expire first this time and R_s will forward the packet prior to the other relays. An ACK packet will be returned if the second-best-relay-channel retransmission succeeds. The same as in Case II, other relay nodes will freeze their timers during the second-best-relay-channel retransmission and reactivate them after ACK2 timeout. As shown in Fig.2(c), the relay nodes will participate in data retransmission consecutively one after another until D decodes the packet successfully and responds with an ACK2 packet. Whenever the ACK2 packet is detected, the remaining relay candidates will reset their timers and discard their received packets, and the cooperative transmission cycle is thus completed.

Case IV: If the cooperation of all relay nodes still does not lead to successful data reception at D, or if the number of retransmission attempt reaches the retry limit, the cooperative transmission fails. As shown in Fig. 2(d), the source node will then obtain channel access again for another round of packet transmission, following the same procedure as described above.

3. SPIN and PROMELA

The terminology for mathematical demonstration of the correctness of a system is *formal verification* [7] and can be accomplished using the SPIN model checker [17]. SPIN is a general

tool for formal verification of distributed software systems. It can be used as a simulator for rapid prototyping with random, guided, or interactive simulations, and it can also be used as an exhaustive verifier proving the validity of user specified correctness requirements.

Models that are analyzed in SPIN must be specified in an internal specification language called PROMELA [7], a verification modeling language providing a way for making abstractions of distributed systems. It attempts to abstract as much as possible from internal sequential computations, focusing on the expression of only essential properties of modeled systems in order to verify whether the process interactions are correct or not. In PROMELA we can specify all essential features of distributed asynchronous systems such as the behavior of nodes or processes (abbreviated as CFSMs), communication channels, and utilize global variables to define the environment in which the processes run. PROMELA allows the dynamic creation of concurrent processes communicating via message channels. Based on a system model specified in PROMELA, SPIN can perform random simulations of the system execution or efficient on-the-fly verification of the system correctness properties [11].

Correctness claims can be specified in the syntax of standard LTL, a model temporal logic with modalities referring to time. Using LTL one can express properties of paths in a computation tree. There are mainly two types of properties that can be expressed in LTL: safety properties (something bad never happens) and liveness properties (something good keeps happening). SPIN does exhaustive search and produces fast programs (called validators), and can be used in different modes. For small to medium size models, the validators can be used with an exhaustive state space. The result of all validations performed in this mode is equivalent to an exhaustive proof of correctness, for the correctness requirements that were specified. For larger systems, the validators can also be used in supertrace mode, with the bit state space technique. In these cases the validations can be performed using much less memory, and still retain excellent coverage of the state space [18].

4. Abstraction and Modeling

We use the SDL language to specify and visualize a formal model for the above presented protocol. SDL can provide us with unambiguous specification and description of the behavior of our system. While it is possible to specify a more generic model for any node (irrespective of its function being Sender, Destination or Relay), we have modeled the behavior separately for better understanding and ease of use. Without losing generality, two relay nodes (Relay R and Relay H) are adopted in our model besides the source node (S) and the destination (D) to illustrate the cooperative procedure. Our model is implemented based on several necessary assumptions:

- *All nodes which are involved in cooperative transmissions can hear each other.*
- *The receiving nodes can always receive the DATA packet from the channel, no matter it is decoded as correct or not.*
- *The ACK and CFC packets are always received correctly.*
- *Between the two relay nodes, R represents the relay node with better relay channel condition hence is chosen to retransmit first.*

The reasons for the above strong assumptions are listed in the following. This novel C-ARQ protocol targets at typical wireless local network scenarios where the involved nodes are one-hop away from each other. It can be an infrastructure network or one-hop neighborhood in an ad hoc network. In an infrastructure work like the current WiFi networks, the nodes are typically densely distributed. The distance between the source and the destination node is usually quite short. In an ad hoc network, a routing protocol will find out the next hop destination. This cooperative scheme will be implemented between the sender and its next hop destination. Therefore, we have the assumption that all the nodes can hear each other and that the ACK and CFC packets will be decoded correctly due to their small packet lengths.

On the other hand, there exist problems like collisions of packets that would cause packet loss in reality. Those problems are inherited from traditional wireless local networks. However, cooperative communications have no contributions towards these problems. This paper focuses on the benefits that cooperative communications can bring in. Therefore, the existing mechanisms like the back-off procedure and the RTS/CTS handshaking procedure, which are introduced to solve the collisions and hidden terminals problems, are not included in our modeling. One reason is that it will for sure dramatically increase the complexity, and another reason is that it might also blur the focus of this study. Given the above considerations, the strict assumptions are made here to simplify the modeling and focus on the cooperative retransmission part of the scheme.

4.1. SDL Model for C-ARQ

The SDL models designed for the sender, S, the destination, D, the optimal relay, R, and second relay H according to the C-ARQ protocol, are shown is Figs. 3 – 6 respectively.

In our models, several virtual signals, such as RTimeout, HTimeout, ACK2Timeout and ACK3Timeout, are designed to be transmitted between processes in order to model the timeout function of the nodes in reality, in addition to the normal packets the nodes receive from the wireless channel, e.g., DATA and ACK packets. All these signals are sent in a broadcast way, and all the other nodes in the network will receive and process the signals according to their own state machines. Note that the backoff procedure in the original DCF scheme is simplified in our model because of its independence from the cooperative protocol.

4.2. PROMELA Implementation

For the simplified system model with one Sender, one Destination and two Relays (R and H), we implemented four separate processes in the same environment in PROMELA, one for each node. While it is possible to implement the system as instances of a single generic node which performs various roles,

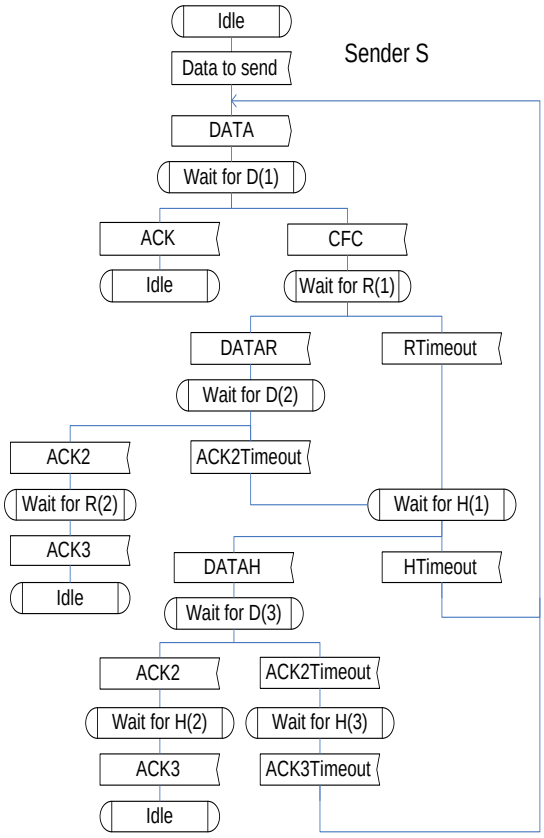


Figure 3: Formal SDL Model for C-ARQ: Sender.

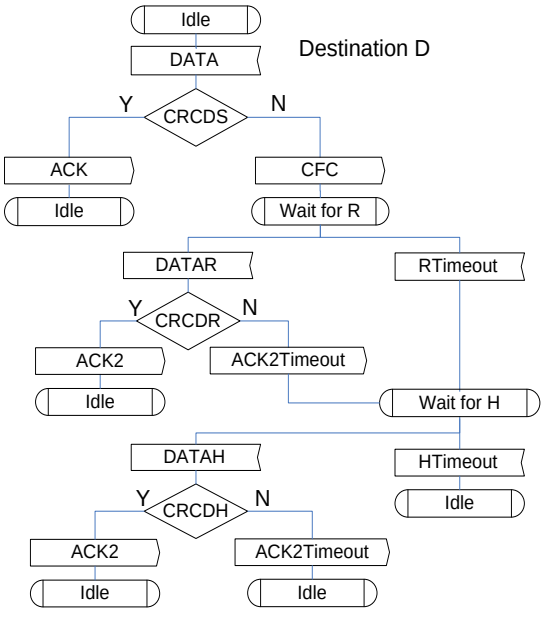


Figure 4: Formal SDL Model for C-ARQ: Destination.

we find it clearer to represent it this way without the loss of protocol logic. In the real world, the messages are exchanged in the wireless channel where every node can receive the messages sent by every other node within its transmission range. For fine grained control over our operating environment, the broadcast

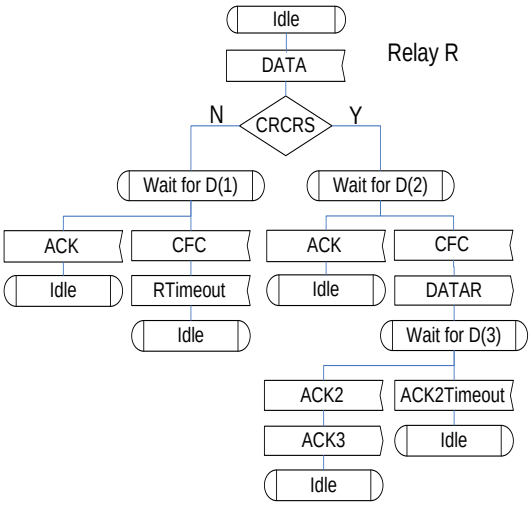


Figure 5: Formal SDL Model for C-ARQ: RelayR.

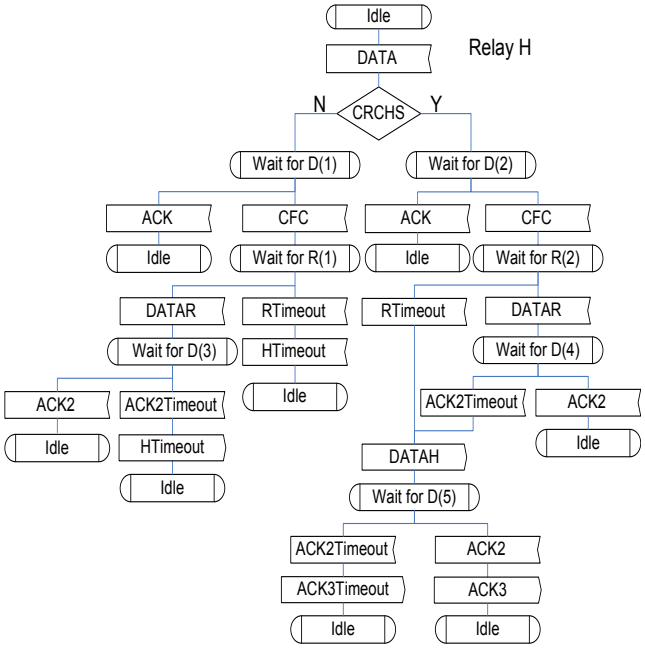


Figure 6: Formal SDL Model for C-ARQ: RelayH.

concept is implemented as six point-to-point synchronous channels between various nodes (see Code:*Channels* below). In this way, we have control over which message needs to be sent and to which node easily, through the corresponding channel.

Code:Channels

```
chan StoD = [0] of {mtype, bool};
chan StoR = [0] of {mtype, bool};
chan RtoD = [0] of {mtype, bool};
chan RtoH = [0] of {mtype, bool};
chan StoH = [0] of {mtype, bool};
chan HtoD = [0] of {mtype, bool};
```

Since the protocol requires information about the correctness of the received DATA packet at various nodes, we define CRC at various nodes as local variables as indicated below

(Code:*CRC variables*):

Code:CRC variables

```
At Sender S: bool CRCDS, CRCRS, CRCHS;
             /*Sent to D, R and H*/
At Relay R: bool CRCDR;/*CRC sent to D*/
At Relay H: bool CRCDH;/*CRC sent to D*/
```

The various conditions of the channel are simulated by setting various CRC variables as *True* or *False* between each transmission pair. The variable configuration and simulation results achieved are described later in Section 5. Different CRC values are set using a random function at its corresponding receiver, which determines the received CRC value to be true or false randomly during each iteration. The random CRC code at the relay, R, from the direct transmission is taken as an example and shown in (Code:*RandomCRC*). Coupled with the infinite packets being sent at the source node, the simulation covers all the possibilities that arise from various permutations of the different CRC values. [1]

Code:RandomCRC

```
/*Random  CRC  at relay R*/
StoR?tmpRS,tmpCRCRS;
  if
     ::tmpRS==data ->
         do
              :: tmpCRCRS=true;  break
              :: tmpCRCRS=false;  break
         od;
         ......
```

Code: Reception at Destination

```
StoD?tmpDS,tmpCRCD;
if
 ::tmpDS==DATA ->
  if
   ::tmpCRCD==true ->
    atomic {StoD!ACK,true; RtoD!ACK,true;
             HtoD!ACK,true;}
         /*Correct Data received*/
   ::tmpCRCD==false ->
    atomic {StoD!CFC,true; RtoD!CFC,true;
            HtoD!CFC,true;}
    RtoD?tmpDR,tmpCRCRD;
   /*Corrupted Data; call for help*/
if
 ::tmpDR==RTimeout ->
  HtoD?tmpDH,tmpCRCDH;
   /*Relay R failed; Wait for Relay H*/
  .....
```

The receiver will respond following the C-ARQ protocol based on the CRC checking results of the received DATA packet. For instance, when the destination D receives DATA from the direct channel, according to the result of CRC checking, ACK or CFC will be sent out to the channel respectively.

[1] Note that this channel configuration method is only used here for protocol verification. Wireless channels in reality are subject to time correlated fading and hence the CRC values cannot be randomly set up in every data transmission iteration. More sophisticated and realistic channel models may be investigated for more comprehensive protocol performance evaluation.

5. Simulation Results

Message generation procedures are simulated in SPIN for all the different cases of the C-ARQ protocol, as described in Section 2. From the simulation results, we can claim that the protocol works exactly as how it is described for all the given cases. The vertical lines in the simulation message sequence charts in this section indicate the Sender, Destination, Relay R (preferred relay) and Relay H respectively.

Case I: *CRCDS = true.*

The direct transmission succeeded without need for retransmissions from relays.

Case II: *CRCDS = false; CRCRS = true; CRCDR = true.*

Relay node R which has decoded the DATA packet correctly forward its DATA packet to the destination successfully after the direct transmission fails.

Case III: *CRCDS = false; CRCRS = true; CRCDR = false; CRCHS = true; CRCDH = true.*

After the optimal relay retransmission fails, the second relay node, H, which also decoded the packet successfully, forward the packet successfully to D.

Case IV: *CRCDS = false; (CRCRS = true; CRCDR = false; CRCHS = true; CRCDH = false) OR (CRCRS = true; CRCDR = false; CRCHS = false) OR (CRCRS = false; CRCHS = false).*

There are three occasions that may cause the failure of the cooperative retransmission after the direct transmission fails: (a) Both R and H have participated forwarding and both packets are corrupted on the relay channels; (b) Only one relay is qualified to participate and the data packet is corrupted; (c) Neither of the two relays have decoded the packet successfully. Hence, no cooperative retransmission happens. In all the three scenarios, the source node S starts to resend the packet after the whole cooperative retransmission procedure fails.

6. Verification using SPIN

6.1. Invalid End-States

In PROMELA, valid end-states are those system states in which every process instance has either reached the end of their defining program body or is blocked at a statement that has a label starting with the prefix *end*. Valid end-states also require channels to be empty. All other states are invalid end-states. In all of our verification operations, no invalid end-states were found.

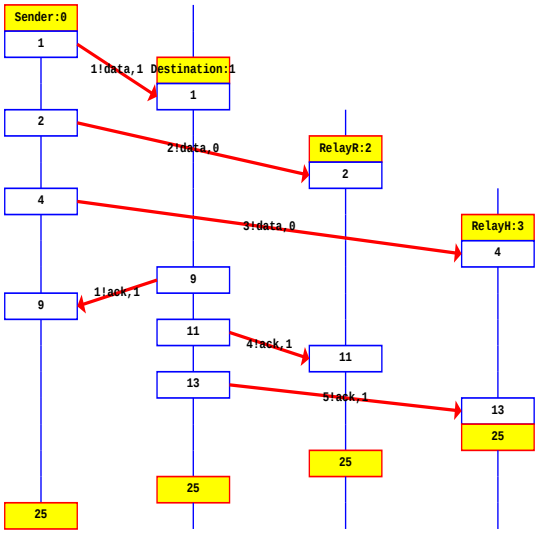


Figure 7: Simulation Result for Case I: Direct Transmission Succeeds.

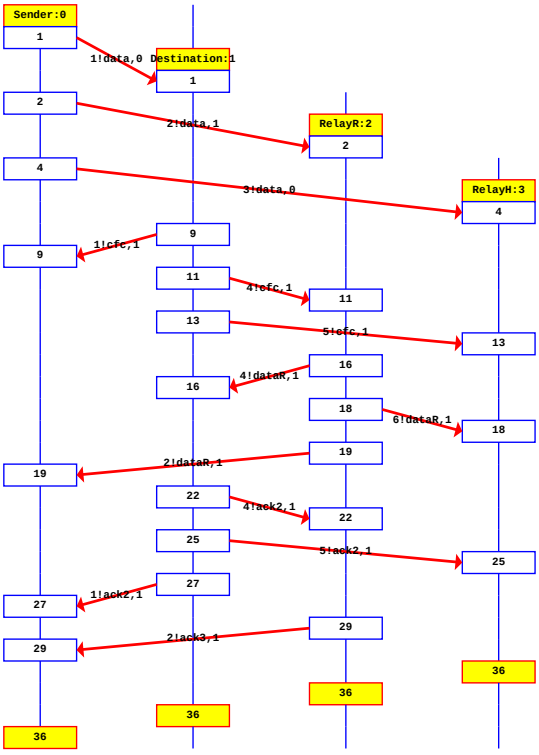


Figure 8: Simulation Result for Case II: Direct Transmission Fails; First Cooperative Retransmission Succeeds.

6.2. *Never Claims*

A straightforward verification of the protocol requirements can be modeled as never-claims using PROMELA. We formalize tasks that are claimed to be performed by the system using the LTL logic. SPIN can either prove or disprove those claims quickly.

In our model, we define the requirement for the C-ARQ protocol as: *If there exists a good data transmission link, either a source-destination link or a source-relay-destination link, the DATA packet should be successfully delivered from source to destination and the ACK packet should be returned to the source.* We formulate the LTL logic for this statement using *SDChannel, SRChannel, RDChannel, SHChannel, HDChannel* and *Transmission* as global mtype variables (See *LTL for Never Claim* below). They are introduced to indicate whether the corresponding channel condition is good (no data corruption) or not and whether the date transmission is successful or not.

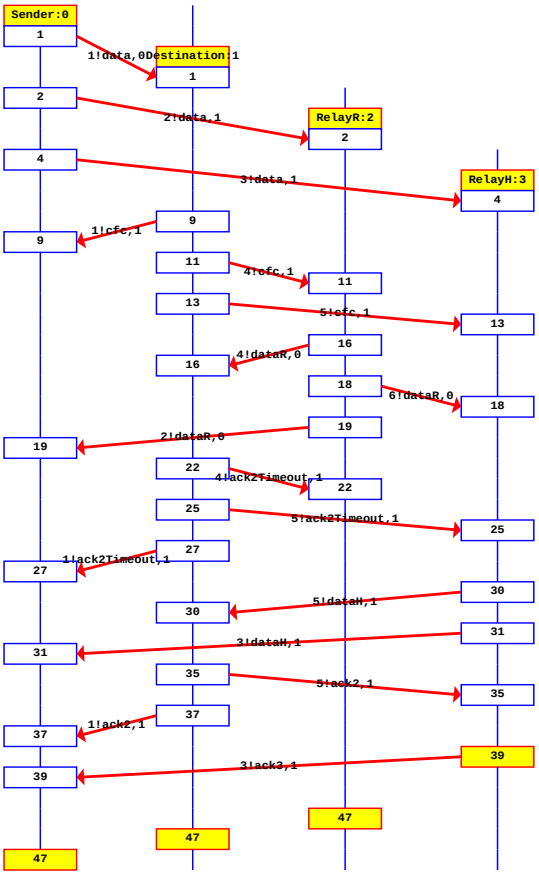


Figure 9: Simulation Result for Case III: Direct Transmission Fails; First Cooperative Retransmission Fails; Second Cooperative Retransmission Succeeds.

LTL for Never Claim

```
[]((psd||(psr && prd)||(psh && phd))
                                -> <> q)
#define psd (SDChannel==good)
#define psr (SRChannel==good)
#define psh (SHChannel==good)
#define prd (RDChannel==good)
#define phd (HDChannel==good)
#define q (Transmission==good)
```

The verification results produced by the SPIN tool are depicted at the bottom of the *Verification output for Never Claim* chart. No errors occurred in the exhaustive verifications, indicating that the claim proposed holds for the C-ARQ protocol.

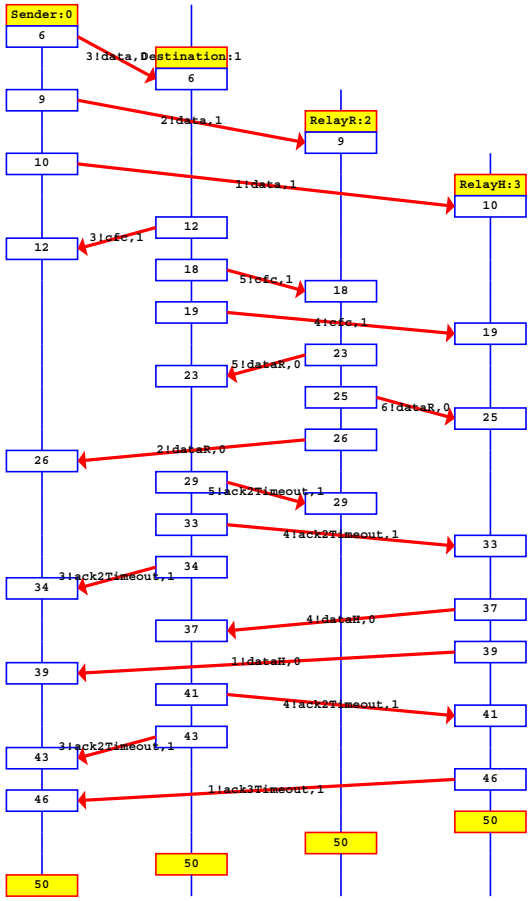


Figure 10: Simulation Result for Case IVa: Cooperative Retransmission Fails: Packet Corruption on Both Qualified Relay Channels.

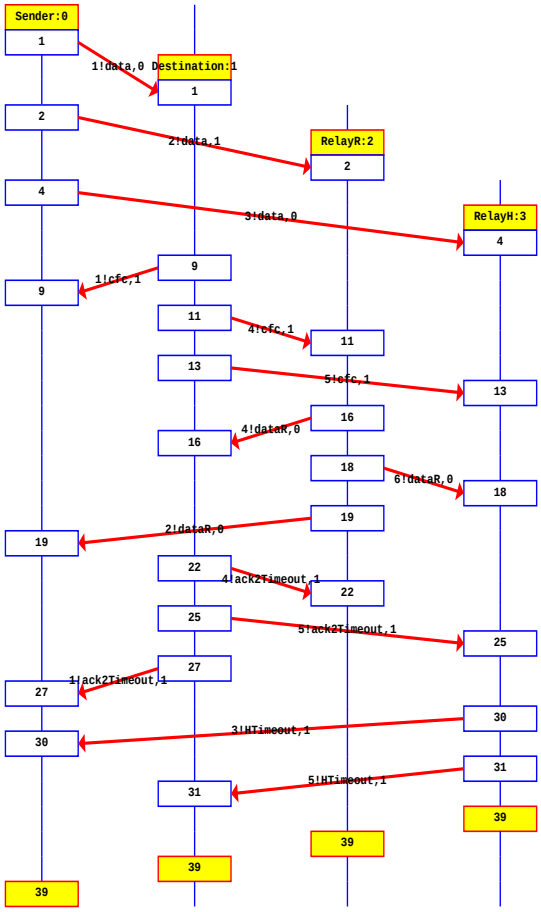


Figure 11: Simulation Result for Case IVb: Cooperative Retransmission Fails: Packet Corruption on Single Qualified Relay Channel.

```
Verification output for Never Claim
warning: for p.o. reduction to be valid
the never claim must be stutter-invariant
 (never claims generated from LTL formula
 are stutter-invariant)
depth 0: Claim reached state 5 line 336)
depth 50: Claim reached state 9(line 341)
depth 48: Claim reached state 9(line 341)
(Spin Version 5.2.5 -- 17 April 2010)
  + Partial Order Reduction
Full statespace search for:
  never claim                   +
  assertion violations          +
              (if within scope of claim)
  acceptance   cycles +
              (fairness disabled)
  invalid end states        -
              (disabled by never claim)
State-vector 116 byte, depth reached 365,
errors: 0
  3739 states, stored (3752 visited)
  931 states, matched
  4683 transitions    (= visited+matched)
  0 atomic steps
hash conflicts:         10 (resolved)
```

7. Discussions

The main contribution of this work is to verify the correctness of the C-ARQ model using simulations and verifications. We have demonstrated that C-ARQ as proposed is indeed correctly executed under various conditions.

To test the protocol using generic descriptions of node instead of specific roles (like sender, receiver or relay) would be overly complex and inefficient. The model can be simplified to a large extent by assigning roles to nodes during a single transaction where (in time) a node can be a sender or a receiver or a relay. The roles can be switched at a later transaction, but our model remains valid since they abstract the roles from the nodes themselves. This approach also enables us to clearly implement and analyze the role specific functions that are needed in this protocol.

To represent the correctness condition for the proposed C-ARQ protocol in LTL is a challenging task. For example, how do we relate the channel conditions (even with packet loss or data corruption) and check if the protocol can function in diverse channel conditions? We accomplish this by randomizing channel conditions in simulations and relating the fundamental conditions to the successful transmission of data. More specifically, the protocol should be able to deliver the data packet

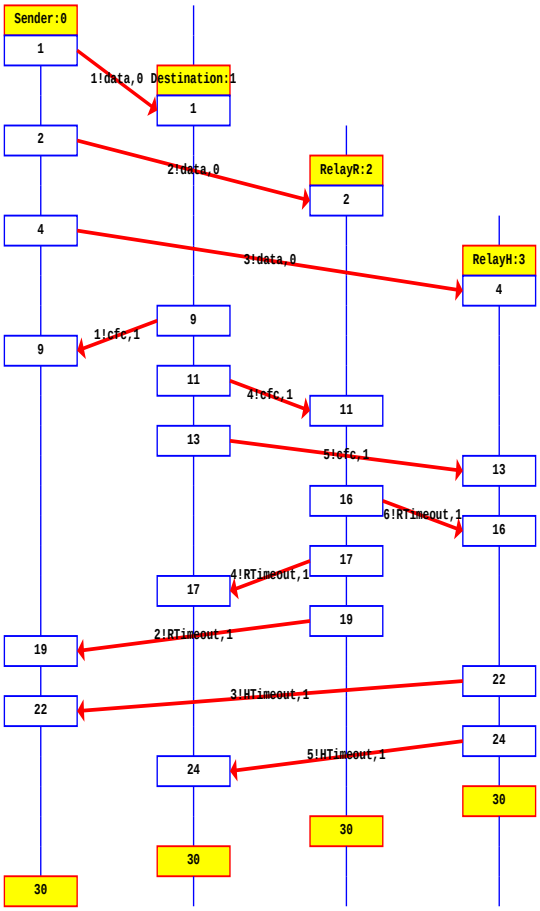


Figure 12: Simulation Result for Case IVc: Cooperative Retransmission Fails: No Available Relay.

correctly if there has been a functional path (direct or indirect via relays) between the source and the destination. See Section 6.1 for the formulation of this condition.

The assumptions and constraints are essential in this verification procedure of the protocol. The verification we have undertaken is an "exhaustive state space mode" which is equivalent to an exhaustive proof of correctness (for the correctness conditions that were specified). In the 'invalid end state' verification (Sec 6.1), we make sure that the protocol never enters a state from which it cannot exit, i.e., avoiding infinite loops (like all nodes waiting for a packet that is never sent or get lost) or race conditions. Further versions of the protocol should satisfy these tests to be able to prove their correctness as well.

8. Conclusions

In this paper, we have described the C-ARQ MAC protocol with a formal SDL model, and used PROMELA along with the SPIN model checker to verify the correctness and the functionality of the protocol. Simulation results from SPIN coincide with the protocol description, and the verifications are carried out through never-claims, using the LTL logic in SPIN. No invalid end-states were found and the proposed claim holds true in the exhaustive verification operations, indicating the integrity and validity of the C-ARQ protocol.

Furthermore, the formal model can be refined with more logic to verify other functions of the protocol. With the basic logic implemented and verified in this paper, further modifications to the model should be straightforward.

Acknowledgments

The authors would like to thank Prof. Andreas Prinz and Prof. Vladimir A. Oleshchuk (both from the University of Agder, Norway) for their valuable feedback and insights in this work.

References

[1] J. Cai, X. Shen, J. W. Mark, Semi-distributed user relaying algorithm for amplify-and-forward wireless relay networks, IEEE Transactions on Wireless Communications 7(4) (2008) 1348-1357.

[2] J. N. Laneman, D. N. C. Tse, G. W. Wornell, Cooperative diversity in wireless networks: efficient protocols and outage behavior. IEEE Transactions on Information Theory 50(12) (2004) 3062-3080.

[3] J. Alonso-Zarate, E. Kartsakli, C. Verikoukis, Persistent RCSMA: A MAC protocol for a distributed cooperative ARQ scheme in wireless networks. EURASIP Journal on Advances in Signal Processing (2008), doi:10.1155/2008/817401.

[4] P. Liu, Z. Tao, S. Narayanan, T. Korakis. CoopMAC: A cooperative MAC for wireless LANs. IEEE Journal on Selected Areas in Communications, 25(2) (2007) 340-354.

[5] J. S. Pathmasuritharam, A. Das, A. K. Gupta, Efficient multirate relaying (EMR) MAC protocol for ad hoc networks. Proceedings of IEEE ICC (2005) 2947-2951.

[6] X. He, F. Y. Li, A multi-relay cooperative automatic repeat request protocol in wireless networks. Proceedings of IEEE ICC (2010) 1-6.

[7] G. J. Holzmann, The model checker SPIN. IEEE Transactions on Software Engineering 23(5) (1997) 279-295.

[8] E. M. Clarke, O. Grumberg, D. Peled (2000). Model checking. ISBN-13: 978-0-262-03270-4. The MIT Press.

[9] IEEE 802 WG, Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE 802.11 (1999).

[10] F. de Renesse, A. H. Aghvami, Formal verification of ad-hoc routing protocols using SPIN model checker. Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference 3(3) (2004) 1177-1182.

[11] V. A. Oleshchuk, Modeling, specification and verification of ad-hoc sensor networks using SPIN. Computer Standards & Interfaces, 28(2) (2005) 159-165.

[12] J. S. Segui, Demand access protocol design and validation with SPIN. IEEE Aerospace Conference (2008) 1-8.

[13] B. Long, J. Dingel, T. C. N. Graham, Experience applying the SPIN model checker to an industrial telecommunications system. Proceedings of the 30th ACM/IEEE International Conference on Software Engineering (2008) 693-702.

[14] J. Li, J. Li, Model checking the SET purchasing process protocol with SPIN. Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing (2009) 1-4.

[15] X. He, F. Y. Li, Cooperative MAC design in multi-hop wireless networks: part I: when source and destination are within the transmission range of each other". Wireless Personal Communications (57)(2010) 339-350.

[16] X. He, F. Y. Li, Optimization of the relay selection scheme in cooperative retransmission networks. Proceedings of IEEE VTC (2011) 1-6.

[17] SPIN tool website, *http://www.spinroot.com*.

[18] G. J. Holzmann, An analysis of bitstate hashing, Formal Methods in System Design 13(3) (1998) 289-307.