# Formal Verification of a Cooperative Automatic Repeat Request MAC Protocol

Xin He, Ram Kumar, Liping Mu and Terje Gjøsæter
{xin.he, ram.kumar, liping.mu, terje.gjosater}@uia.no

**Abstract**

Cooperative communications, in which a relay node assists the source node to deliver its packets to the destination node thus adding diversity, is gaining more attention within the research community. The initial studies on the physical layer have shown significant gains from cooperative diversity in terms of reliability, coverage range and energy efficiency. A Cooperative Automatic Repeat reQuest MAC protocol, (*C-ARQ*), has been proposed to utilize cooperative diversity from the MAC layer. In this paper, we inspect this novel *C-ARQ* protocol to check its integrity and validity using formal methods. The protocol logic is modeled in SDL and implemented in PROMELA. The functionality of the C-ARQ protocol is confirmed through simulations and verified using the SPIN tool.

## 1 Introduction

The reliability of any communication system depends heavily on the channel it is operating on and also the physical and Media Access Control layer protocols being used. The Media Access Control (MAC) is often said to be a sub-layer of the OSI data Link layer, which provides the means to access the the physical medium used for communication. Adding time and spacial diversity to physical and MAC layer protocols provide additional steps to mitigate various channel fading. Cooperative communications, which are proposed as a distributed approach to achieve time and space diversity in wireless environments, is increasingly gaining attention within the research community. In cooperative communications, the information is not only transmitted through a direct link from source to destination but the same DATA packet may also be forwarded by one or more relay nodes to the destination node. The theory behind cooperation has been studied in depth and significant improvement of system performance has been demonstrated in terms of throughput, network coverage and energy efficiency [1].

Much study is being done on cooperative Medium Access Control (MAC) design in distributed wireless networks [2, 3]. Three key issues need to be addressed within this perspective: when to cooperate, whom to cooperate with and how to protect cooperative transmissions. A **C**ooperative **A**utomatic **R**epeat re**Q**uest protocol (C-ARQ) has been proposed [4], in which the aforementioned three issues concerning cooperative transmissions at the MAC layer are handled efficiently with a minimum cost of resources.

*This paper was presented at the NIK-2010 conference; see http://www.nik.no/.*

Primarily, 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 is 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 and to protect ongoing packet forwarding by relay nodes.

One mature approach for ensuring reliability in algorithms and protocols 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 [5,6]. 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. The advantage of this approach is that extremely large state-spaces can often be traversed in minutes.

In this paper, we use formal approaches to model and verify the C-ARQ protocol. In section 2, we examine the proposed system model and the C-ARQ protocol in detail. In section 3, we introduce the formal methods selected for use in this article. In further sections, we describe the SDL model for our protocol (section 4), and explain the simulation and verification results (section 5 and section 6) from the SPIN tool. We conclude and discuss the results in section 7.

# 2 C-ARQ Protocol Description

## System Model for Protocol Description

Our cooperative protocol is proposed based on the Carrier Sense Multiple Access With Collision Avoidance (CSMA/CA) scheme in a single-hop scenario, where the source and the destination can hear each other, and there is only one node sending packets at one time. Hence, we use the network in Fig.1 for illustrating the cooperative protocol procedure [4]. The network consists of a source node, S, a destination node, D, and several other randomly distributed potential relay nodes, $R_1$, $R_2$, ... $R_n$.

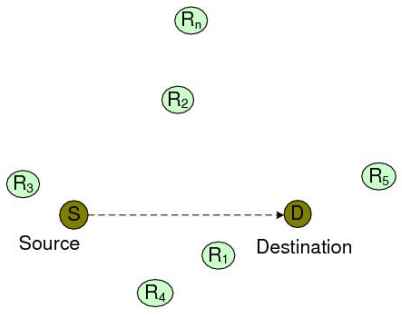



Figure 1: System Model for Cooperative Transmission.

In 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 our proposed protocol.

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 an virtual node in the whole transmission route.

## C-ARQ MAC Protocol Description

The message exchange sequence of the C-ARQ basic access scheme is illustrated in Fig.2 [4]. It has four operation cases, as I) direct transmission succeeds; II) best-relay-channel retransmission succeeds; III) multi-relay retransmission succeeds; and IV) the whole cooperative retransmission fails. In more details, it works as follows.

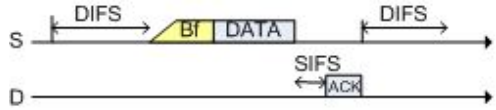
**Case I):** As the first step, the source node S sends out a DATA packet to its destination D following the original Distributed Coordination Function (DCF) basic access scheme [7]. According to the DCF protocol shown in Fig.2(a), S listens to the channel for a DCF InterFrame Space (DIFS) interval and 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 a 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 have 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) [4].

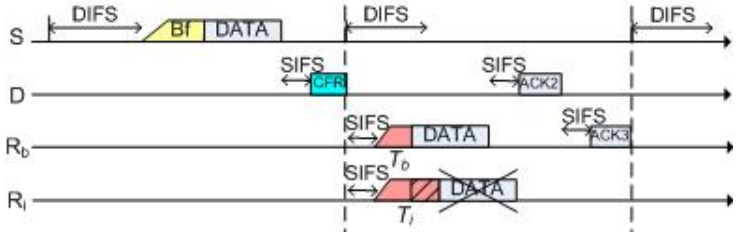
$$T_i = \left\lfloor \frac{SNR_{low}}{SNR_i} \frac{DIFS - SIFS}{slottime} \right\rfloor, \quad i = 1, 2 \cdots n \tag{1}$$

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 $R_i$, and $SNR_{low}$ is the threshold of $SNR_i$ for $R_i$ to participate in the cooperative retransmission; $n$ is the number of the 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 network. The granularity of $T_i$ is specified to be *slottime* of the system in order to cover the propagation delay in the network.

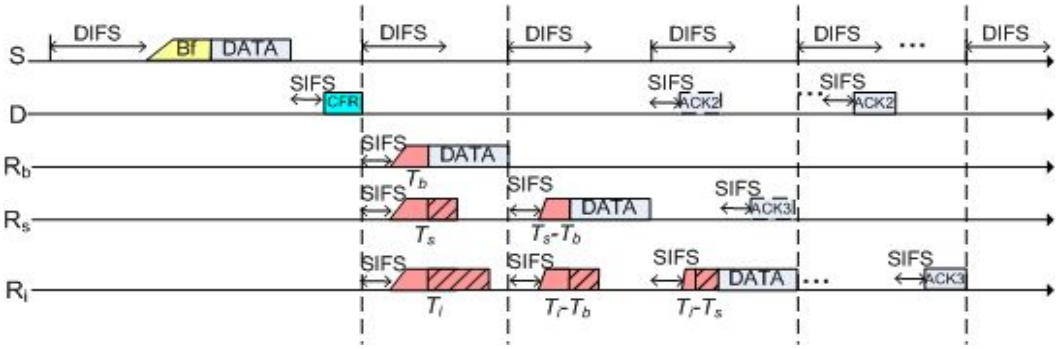
According to Eq.(1), the relay node with the best relay channel quality $R_b$, which observes the strongest received signal and thus has the shortest backoff time $T_b$, will first get access to the channel and forward its received packet to the destination. When the other relay candidates hear the packet 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 [4].



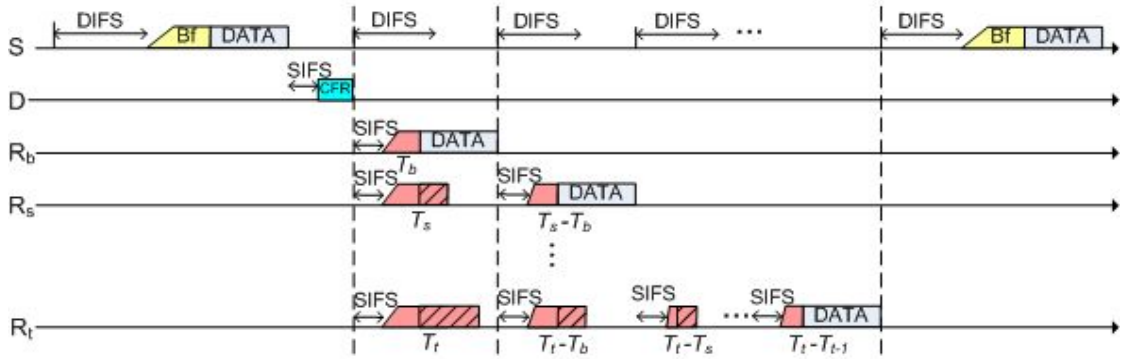

(a) Case I: Direct Transmission Succeeds




(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

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):** The cooperative retransmission failure can be caused either by collisions among two or more relay nodes with the same 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 before the other relays. An ACK packet will be returned if the second-best-relay-channel retransmission succeeds. 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 continually 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 provide a successful data decoding 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 obtain channel access again for another round of packet transmission if the retransmission through the last relay node $R_t$ fails.

# 3 Model Checking

Using model checking technique to formally verify communication protocols has been widely used. Related work can be found in [8–11]. Analyzing a protocol with model checking consists of primarily constructing an abstract description, or model, of the protocol with the main features that could produce execution errors. The reliability properties 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.

## SPIN and PROMELA

The term for this mathematical demonstration of the correctness of a system is formal verification [5] and can be accomplished using the SPIN model checker [12]. 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; it can be used as an exhaustive verifier proving the validity of user specified correctness requirements. The description of the system is the input to the tool (the possible behavior) along with the requirements (the desirable behavior) [6]. Knowing these parameters, the tool can perform a verification of the model. To facilitate the detection of errors, the tool provides a counter example showing under which circumstances the errors can occur. This process can be done iteratively to check all the desired characteristics.

This model checker accepts design specifications written in PROMELA (Process or Protocol Meta Language) [5], a verification modeling language providing a way for making abstractions of distributed systems. Correctness claims can be specified in the syntax of standard Linear Temporal Logic (LTL), a modal temporal logic with modalities referring to time. Using LTL one can express properties of paths in a computation tree and 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) which can be used further 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 [9]. In these cases the validations can be performed using much less memory, and still retain excellent coverage of the state space [13].

# 4 Abstraction and Modeling

We use SDL to specify and visualize a formal model for the above mentioned system. SDL can provide us unambiguous specification and description of the behaviour of our system. While it is possible to specify a more generic model to model the nodes (irrespective of their 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 (R and 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 the nodes can hear each other.*
- *The receiving nodes can always receive the DATA packet from the channel (even though erroneously in poor channel condition).*
- *The Acknowledge packets and the CFC packet are always received correctly due to their small packet size.*
- *Between the two relay nodes, R represents the relay node with better relay channel condition hence is chosen to retransmit first.*

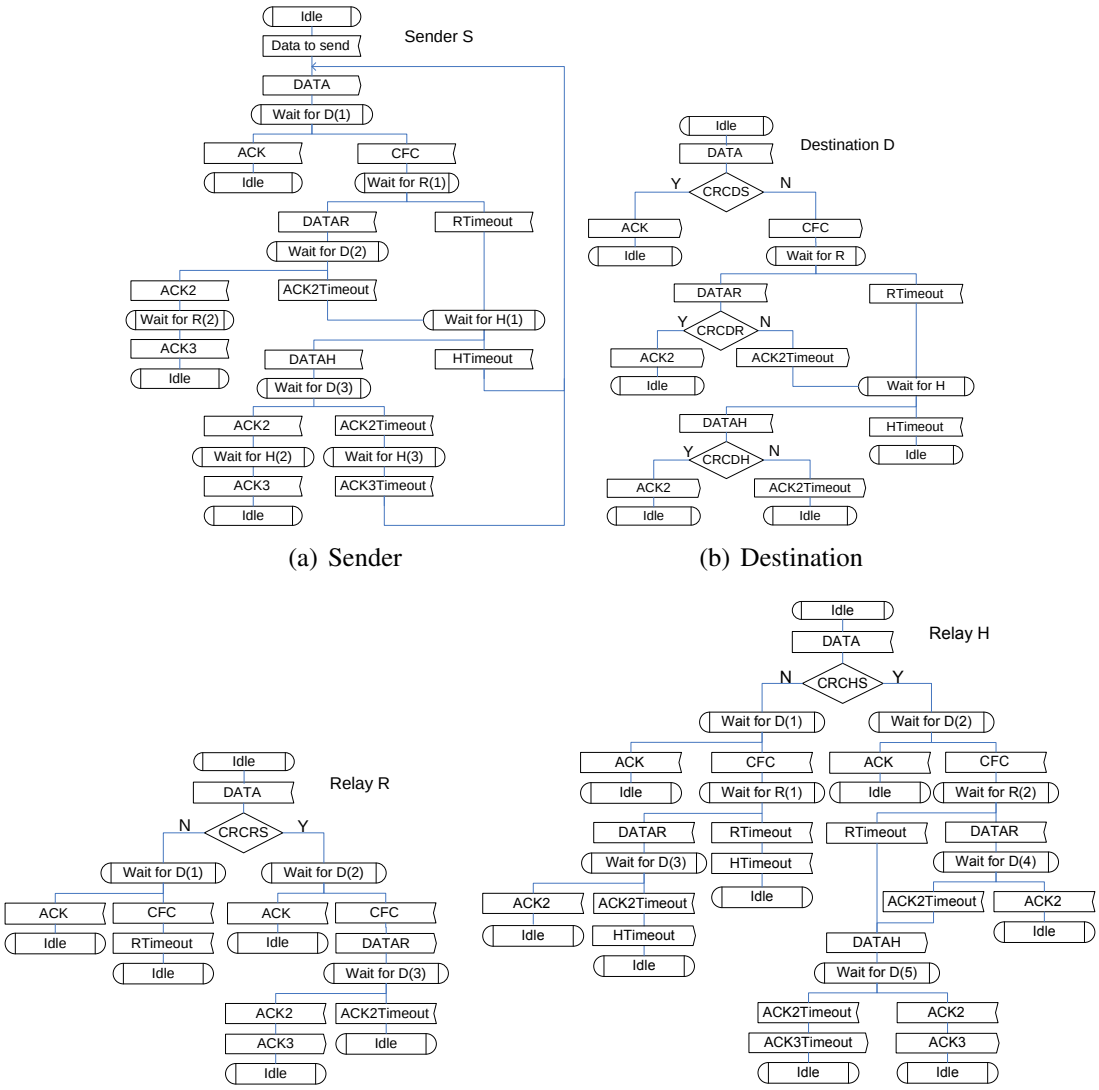
## SDL Model for the CARQ Protocol

The SDL models designed for the sender S, destination D, optimal relay R and second relay H according to the C-ARQ protocol, are shown is Fig.3(a), Fig.3(b), Fig.3(c) and Fig.3(d) respectively.

In our models, besides the normal packets the nodes receive from the wireless channel, for example, DATA and ACK packets, 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. 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.

## PROMELA Implementation

For the simplified system model with one Sender, one Destination and two Relays (Relay R and Relay 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 assumes the various roles, we found 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




(a) Sender (b) Destination

(c) Relay R (d) Relay H

Figure 3: Formal SDL model for C-ARQ protocol

environment, we implemented the broadcast concept as 6 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 by using the channels.

Code:Channels

```
/*Channels for Source, Destination and Single Relay*/
chan StoD = [0] of {mtype, bool};
chan StoR = [0] of {mtype, bool};
chan RtoD = [0] of {mtype, bool};
/*Channels to accommodate Relay H*/
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 the various CRC variables *True* or *False* between each transmission pair. The variable configuration and simulation results achieved are described in section 5. The different CRC values are randomly 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 situations that arise from various permutations of the different CRC values. 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 model should be investigated for protocol performance evaluation.

Code:RandomCRC

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

Code: Destination responses to the received DATA packet

```
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 according to the protocol upon different results of CRC checking 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.

# 5 Simulation Results

Messages sending 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 see that the protocol works exactly as how it is described for all the given cases.

**Case I:** *CRCDS = true.*

The direct transmission succeeded without the help of relays.

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

The 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

## 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 are blocked at a statement that has a label that starts with the prefix *end*. Valid end-states also require channels to be empty. All other states are invalid end-states. In our verification runs, we found no invalid end-states.

## 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 Linear Time Temporal logic (LTL) formula. Spin can quickly either prove or disprove that claim.

In our model, we define the requirement for the CARQ protocol to be: *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 *Transmission* and *SDChannel, SRChannel, RDChannel, SHChannel, HDChannel* as global mtype variables (See *LTL for Never Claim* below). They are introduced to indicate whether the date transmission is successful or not and whether the corresponding channel condition is good (no data corruption) or not.

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 SPIN tool is presented after that in *Verification output for Never Claim*. No errors in the exhaustive verifications, indicate that the claim proposed holds for this C-ARQ protocol.

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
 formulae 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 Conclusions

In this paper, we introduce a SDL model for the the cooperative MAC protocol, C-ARQ, and use PROMELA along with the SPIN tool to verify the protocol. Four Processes are generated: source S, destination D, optimal relay R and second relay H. The cooperative retransmission only happens when the direct transmission fails, and the second relay is only adopted when the optimal relay fails. Simulation results from SPIN coincide with the protocol description, and the verifications are carried out through never-claims, using LTL formula in SPIN.

We could further build and refine the protocol to include more logic that might expand the scope of this protocol in the future, together with the procedure covered in this paper

to verify the additional functionalities as the protocol is expanded. As we already have a basic logic implemented, working and verified in this paper, further modifications to the model should be straightforward for anyone who wants to build on these results.

## Acknowledgements

The authors would like to thank Prof. Andreas Prinz (University of Agder, Norway) for his valuable feedback and insights related to this article.

## References

[1] Laneman, J. N., Tse, D. N. C., & Wornell, G. W. (2004). Cooperative diversity in wireless networks: efficient protocols and outage behavior. *IEEE Trans. Inform. Theory*, 50(12): 3062-3080.

[2] Alonso-Zarate, J., Kartsakli, E., Verikoukis, C., & Alonso, L. (2008). Persistent RCSMA: A MAC protocol for a distributed cooperative ARQ scheme in wireless networks. *EURASIP Journal on Advances in Signal Processing*, 13.

[3] Liu, P., Tao, Z., Narayanan, S., Korakis, T., & Panwar, S. (2007). CoopMAC: A cooperative MAC for wireless LANs. *IEEE JSAC*, 25: 340-354.

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

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

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

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

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

[9] J.S. Segui. (2008). Demand Access Protocol Design and Validation with SPIN. *IEEE Aerospace Conference*, 1-8.

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

[11] Li Jing., & Li Jinhua. (2009). Model Checking the SET Purchasing Process Protocol with SPIN. *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 1-4.

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

[13] Holzmann, Gerard J. (1995). An Analysis of Bitstate Hashing. In *Form. Methods Syst. Des*.