Sindh Univ. Res. Jour. (Sci. Ser.) Vol.43 (1-A) 14-20 (2011)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Model-Checking Techniques for Reliable Communication over Unreliable Media

Saleem Vighio and Anders P. Ravn

Department of Computer Science,Selma Lagerlöfs Vej 300, DK-9220, Aalborg East, Denmark.

vighio@cs.aau.dk, apr@cs.aau.dk

*Corresponding authors:
receive

Abstract: We present a simple approach to modeling and verification of the well-known Alternating Bit Protocol using the model checker UPPAAL to investigate different communication assumptions of the protocol, for instance that the messages can be lost or corrupted during the transmission. We verify safety and liveness properties of the protocol for reliable communication and ensure that the protocol satisfies these properties even in the case of message failures.

Keywords: Communication Policies; Model-Checking; Verification.

Achieving reliable communication over unreliable channels is always a challenge for the designers of communication protocols, see (Holzmann, 1955) and (Afek, *et al*., 1994). Therefore, protocol verification relies on assumptions about the underlying communication media. In this paper, we present model checking techniques which illustrate this issue by modeling various media in a verification of the Alternating Bit Protocol (Bartlett. *et al*., 1969) (Lynch, 1968).

The Alternating Bit Protocol (ABP for short) is a well-known simple data link layer protocol used as a test case for the analyses of concurrent systems, see for example (Bochmann, 1995) (Bergstra, *et al*., 1986).

In a similar work, the ABP is analyzed in (Cleaveland, 1993) using the Concurrency Workbench (Cleaveland, *et al*., 1993). The authors in (Cleaveland, 1993), present several models of lossy and safe media and verify the correctness of the protocol when some specific model of the medium adopted. However, they do not consider message re-transmissions if the messages are lost during transmission. Since we are using timed automata in our models, we can extend their work, so that, lost messages are re-transmitted if they are not acknowledged after some time delay. Furthermore, in order to ensure that message re-transmissions do not happen infinitely often, we give a time bound on each lossy period. This ensures liveness of the protocol (Ravn, *et al*., 2011).

We consider different communication media types for our analyses such that data and/or acknowledgements can be lost or corrupted during the transmission. We systematically verify and improve the protocol models to ensure that the protocol is correct in its behavior even in the case of (data and/or acknowledgement) failures (loss or corruption). For this purpose, we verify safety and liveness properties of the protocol, whose satisfaction ensures the correct operation of the protocol.

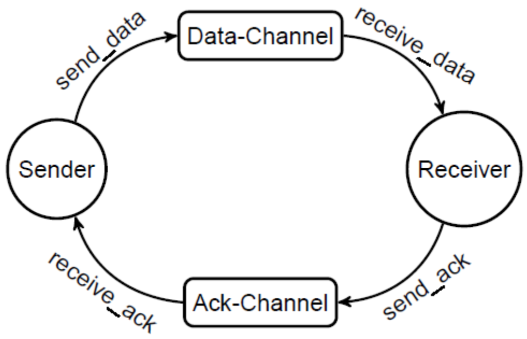


Fig. 1. Alternating Bit Protocol Diagram

The rest of the paper is organized as follows. In Section II, we give an overview of the ABP and introduce the model checker UPPAAL. Section III, provides UPPAAL models of the ABP. Protocol properties and their corresponding UPPAAL semantics is discussed in Section IV. Moving to Section V, we discuss in detail and verify different communication media types of the ABP. Finally, Section VI gives a summary of verification results and concludes with a discussion of the results and also suggests future work.

MATERIAL AND METHODS

A. *The Alternating Bit Protocol*

The ABP is a simple data link layer network protocol often used as a test case for the analysis or verification of concurrent systems. **(Fig. 1)** shows a control flow diagram of the ABP.

The protocol consists of three roles: A *Sender*, a *Receiver*, and a medium. In Fig. 1, we consider full-duplex communication therefore, we divide medium into *Data-Channel* used to receive and send data between the *Sender* and the *Receiver* respectively, and *Ack-Channel* used for receiving and sending acknowledgements between the *Receiver* and the *Sender* respectively. Note that these channels could be used concurrently in opposite roles to achieve a full duplex communication.

We model and verify the ABP using the model checker UPPAAL. UPPAAL is an integrated tool environment for modeling, simulation and verification of real-time systems, modeled as a parallel composition of non-deterministic timed automata (Behrmann, *et al*., 2004). A timed automaton in UPPAAL is a finite state machine that can be extended with real-valued clock variables, discrete variables ranging over finite domains, invariants on locations and guards on transitions. Guards and locations may contain both discrete valued variables and real-valued clocks. Several automata running in parallel can synchronize on transitions by means of channel symbols and can also transfer data with the use of shared variables. Channels can be urgent, which does not allow any delays on synchronizations. Locations can be urgent or committed. An urgent location does not allow time to progress and a committed location restricts interleaving, which means that transitions through a committed location are taken as one atomic step.

UPPAAL uses a subset of CTL (Computation Tree Logic) as a query language to verify properties of systems. The tool can verify safety, reachability and liveness properties formulated in its query language.

B. *Modeliing of ABP in UPPAAL*

- *Sender*

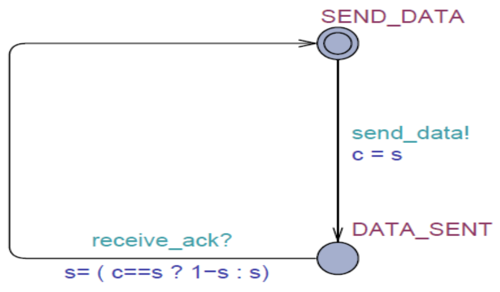


Fig. 2. Sender

Initially, the *Sender* sends a message, which here is abstracted to a control bit, along the medium. The *Sender* continues sending the message until it receives an acknowledgement of a successful transmission. After a successful transmission, the *Sender* flips the control bit and starts all over again. The UPPAAL model for the *Sender* is shown in **(Fig. 2)**. The model contains two urgent channels *send_data* and *receive_ack* synchronizing with the medium in order to send data and to receive acknowledgements respectively. It uses a global integer variable *c*, which is used to exchange the control bit values; local integer variable *s*, stores the control bit.

Once the *Sender* has received an acknowledgement of sent data, it flips the control bit (*s= (c==s ? 1-s : s))* and sends a new frame. The *Sender* in **(Fig. 2)** does not resend data, however, Figure 6 shows a situation where the *Sender* resends data after some time delay and keeps sending data until it receives an acknowledgement.

- *Receiver*

The *Receiver* reads a frame from the medium and sends an acknowledgement, which is the control bit of the received message.

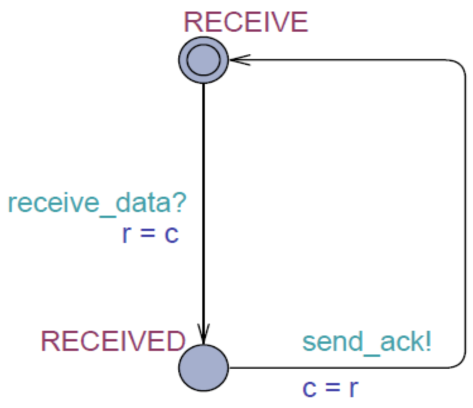


Fig. 3. Receiver

The UPPAAL model for the *Receiver* is shown is Figure 3. It consists of two urgent channels *receive_data* and *send_ack*, used to synchronize with the medium in order to receive data and to send acknowledgements of received data respectively. It also uses the global integer variable *c* to exchange the control bit value and uses a local integer variable *r* to store control bit values.

- *Medium*

The medium is responsible for transmitting frames. We define two instances of the medium. One dealing with data, and the other dealing with

First author is supported by Quaid-e-Awam University of Engineering, Science, and Technology, Nawabshah, Pakistan and partially by the Nordunet3 project COSoDIS.

acknowledgements. The UPPAAL instantiations for the *Sender*, the *Receiver* and the media are coded as

sender = Sender(send_data, receive_ack);
receiver = Receiver(receive_data, send_ack);
data_medium = Medium(send_data, receive_data);
ack_medium = Medium(send_ack, receive_ack);

where *send_data*, *receive_data*, *send_ack*, and *receive_ack* are declared as urgent channels before the process instantiations. The system configuration for the *Sender*, the *Receiver* and media are coded in UPPAAL as

System sender, receiver, data_medium, ack_medium;

The model checker UPPAAL has C-based syntax and allows parameterization of process templates, for e.g., process template for Medium can be parameterized as

Medium(urgent chan &send, urgent chan &receive)

where channel *send* corresponds to *send_data* and *send_ack*, passed as reference parameters from the processes for *data_medium* and *ack_medium* respectively, and channel *receive* corresponds to *receive_data* and *receive_ack*, passed as reference parameters from the processes for *data_medium* and *ack_medium* respectively.

C. Protocol Properties

In this section, we discuss properties a correct protocol needs to satisfy. As stated in (Yemini, *et al*., 1982), verification is usually divided into proving safety and liveness properties. A safety property asserts that nothing bad will ever happen. A liveness property on the other hand asserts that something good will eventually be achieved.

The most basic safety property of a communication protocol is freedom from deadlocks. This property is easily formulated in the UPPAAL query language as

A [] not deadlock *(1)*

However, satisfaction of this safety property does not prove that the protocol transfers data, it may lack progress. Therefore, we verify two liveness properties of the protocol. The first liveness property written in UPPAAL query language below, ensures progress in the *Receiver*, and thus indirectly in the *Sender*. We call this property as local progress.

receiver.RECEIVED and receiver.r==s) - - >
receiver.RECEIVED and receiver.r==1-s) *(2)*

We verify property (2) for both cases when initially *s* is *0* and when *s* is initially *1*. Property (2) states that, if the *Receiver* (shown in Figure 3) in location *RECEIVED* has received a frame (from the *Sender* along the medium) with the sequence number *s* then the next frame it will receive has the sequence number *1-s*. It ensures progress in the behavior of the ABP such that after a successful transmission of data the *Sender* flips the control bit and starts sending a new frame, the ABP.

The second liveness property (property (3), shown below) ensures progress not only at the *Receiver* side but also ensures progress at the *Sender* side. We call this property as global progress. This property ensures that the *Sender* and the *Receiver* are synchronized such that data (the control bit) sent is also received. The UPPAAL formulation of this property is shown below:

((sender.DATA_SENT and sender.s == s) and
((receiver.RECEIVE and receiver.r == s) or
(receiver.RECEIVED and receiver.r == s))) - - >
((sender.DATA_SENT and sender.s == 1-s) and
((receiver.RECEIVE and receiver.r == 1-s) or
(receiver.RECEIVED and receiver.r == 1-s))) *(3)*

We verify this property for symmetric cases, when initially *s* is *0* and when *s* is initially *1*. This property states that, if the *Sender* has already sent a frame with sequence number *s* and the *Receiver* has already received the same control signal in all its locations then the next frame the *Sender* will send has sequence number *1-s* and the *Receiver* will also update its internal local variable *r* with the sequence number *1-s*. The satisfaction of this property ensures progress not only in the behavior of the protocol but also ensures that there is no inconsistency such that the *Sender* sends a control signal *s* and the *Receiver* receives a control signal *1-s* and vice versa.

We verify these properties for a number of communication media models with varying fault modes.

<u>**RESULTS DISCUSSION**</u>

There are three situations that can occur. The frame is properly transmitted, it is lost, or it is corrupted during the transmission. In this Section, we discuss and implement several implementations of the medium where data and/or acknowledgements can be lost or can be corrupted. We subsequently verify to what extent ABP implements a reliable communication even in the case of messages loss or errors.

Perfect Medium

Perfect medium always transmits data. The UPPAAL model for the perfect medium **(Fig. 4)** contains two urgent channels *send* and *receive* to receive and to send messages (data or acknowledgements) respectively. The model uses one global integer variable *c* to exchange values and one local integer variable *m* is a one place buffer to store values.

The protocol model with perfect medium satisfies the safety property and both liveness properties because no communication failures (loss or corruption) of data or acknowledgements occur.

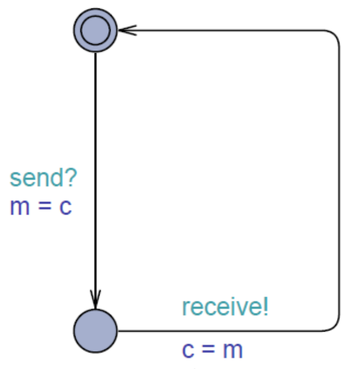


Fig. 4. Perfect Medium

D. *Lossy Medium*

In this mode of communication, data transmitted by the *Sender* and/or acknowledgements transmitted by the *Receiver* can be lost by the medium. Therefore data and/or acknowledgements do not necessarily reach their destination. The end result is that the *Sender* can wait for the acknowledgements forever or the *Receiver* never receives anything.

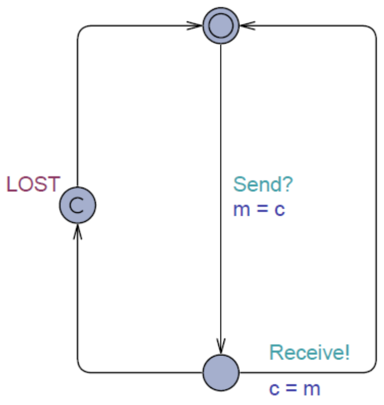


Fig. 5. Lossy Medium

The model for the lossy medium is shown in **(Fig. 5)**. An infinitely long wait for acknowledgement leads the protocol to a deadlock situation and thus violation of the safety property. However, this problem can be solved by introducing data retransmission at the *Sender*, if the *Sender* does not receive any acknowledgement after some specific time delay. **(Fig. 6)**, shows an UPPAAL model of the *Sender* implementing data retransmission after a time delay.

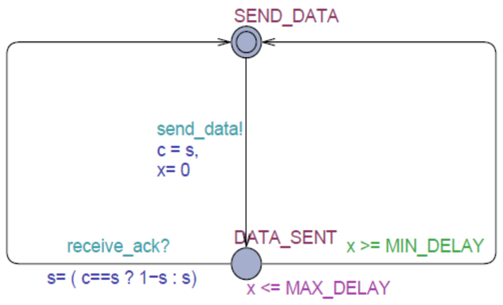


Fig. 6. Sender Data Retransmission

The local clock variable *x* is used to impose progress and two local integer constants *MAX_DELAY* and *MIN_DELAY* are used as a state-invariant *x<=MAX_DELAY* and as a transition-guard *x>=MIN_DELAY* respectively. When the time reaches *MIN_DELAY* before any acknowledgement is received then the *Sender* is ready to retransmit the data again and it must retransmit it before *MAX_DELAY* time units have elapsed. Verification of the safety property is satisfied with this *Sender*. The *Receiver* will not deadlock, because retransmission ensures that it is live, and lost acknowledgements are the concern of the *Sender*. However, liveness is not satisfied. This happens because messages (data or acknowledgements) can be lost infinitely many times so there is no guarantee that the *Sender* will eventually succeed to receive an acknowledgement. We solve this problem with a simple technique of bounding with time, data (and/or acknowledgement) can be lost.

Data retransmission and time-bounded lossy medium

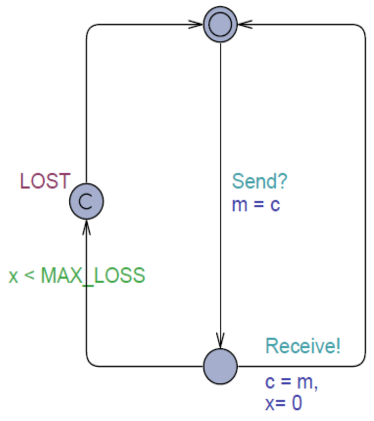


Fig. 7. Time-bounded Lossy Medium

The UPPAAL model for time-bounded lossy medium **(Fig. 7)** bounds to a time (*x< MAX_LOSS*) messages (data and/or acknowledgements) can be lost. The *x* in the guard *x<MAX_LOSS* is a clock, local to the medium and *MAX_LOSS* is a local integer constant. The guard restricts the medium to lose messages to *MAX_LOSS* times and after this time the medium must transmit at least one message to the destination *Sender*/*Receiver*.

The safety and liveness properties are satisfied under this communication type. This gives a natural model of the ABP which ensures correct operation even in the case of intermittent data and/or acknowledgements loss. However, there is yet another possibility where data or acknowledgements can be corrupted but not lost. In this case the *Sender* resends the data without the need of a time out (which was the case with lossy medium) because data and/or acknowledgements are always received in this case though corrupted. It is obvious that the safety property for such a communication discipline is always satisfied as there are no deadlocks. However, the liveness property is only assured when we bound to a time or to a count the number of times the corrupted data and/or acknowledgements are transmitted. This is analogous to the case of the lossy medium. **(Fig. 8)** gives a summary of verification results of safety and liveness properties for all discussed cases.

CONCLUSION AND FURTHER WORK

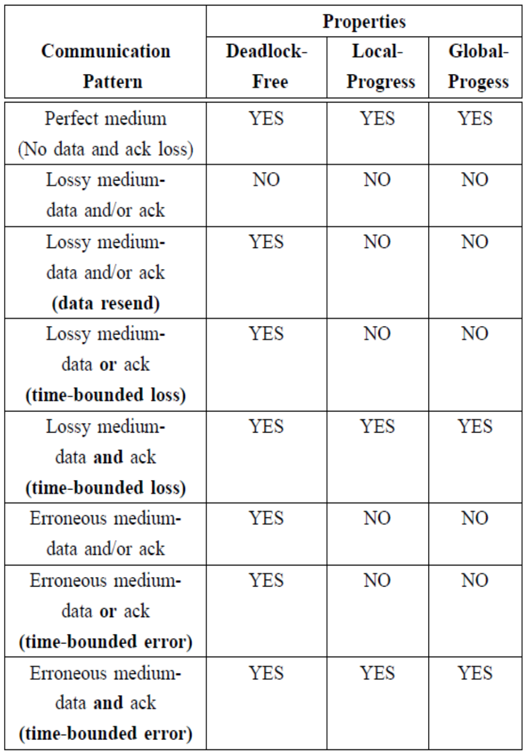
In this paper, we presented a simple approach to modeling and verification of the Alternating Bit Protocol. The Alternating Bit Protocol like other communication protocols allows messages to be lost or corrupted during the transmission. We consider and model possible communication policies of the protocol and systematically verify them to ensure correct operation of the protocol using the model checker UPPAAL. The correctness of the protocol is ensured with the satisfaction of safety and liveness properties, which we call deadlock-free property, local progress, and global progress properties respectively. **(Fig. 8)**, shows verification results of these properties for all discussed communication policies. Our modeling approach also considers a case where lost messages are re-transmitted after some time. We also ensure that, re-transmissions should not happen infinitely often which may lead the protocol to a behavior without progress. Therefore, we restrict the time the messages can be lost so that the protocol can eventually make progress. Our verification results show that the protocol ensures a reliable communication even in the case of message(data and/or acknowledgement) failures(loss or corruption). This is proved in our verification results, when the medium does not allow message loss and corruption, or when data and acknowledgement loss or corruption is time-bounded.

ACKNOWLEDGMENT – I

We are very grateful to Jiri Srba for instructive comments on the text.

ACKNOWLEDGEMENT – II

This is an extended version of a paper presented and published as Conference proceedings in "The International Conference on Computers and Emerging Technologies" (ICCET 2011) held on 22-23 April 2011 at Shah Abdul Latif University, Khairpur, Sindh, Pakistan.



Communication Pattern	Properties		
	Deadlock-Free	Local-Progress	Global-Progess
Perfect medium (No data and ack loss)	YES	YES	YES
Lossy medium- data and/or ack	NO	NO	NO
Lossy medium- data and/or ack **(data resend)**	YES	NO	NO
Lossy medium- data **or** ack **(time-bounded loss)**	YES	NO	NO
Lossy medium- data **and** ack **(time-bounded loss)**	YES	YES	YES
Erroneous medium- data and/or ack	YES	NO	NO
Erroneous medium- data **or** ack **(time-bounded error)**	YES	NO	NO
Erroneous medium- data **and** ack **(time-bounded error)**	YES	YES	YES

Fig. 8. Verification Results of Safety and Liveness Properties for Different Communication Patterns of the ABP.

REFERENCES

Bartlett, K. A., R. A. Scantlebury, and P. T. Wilkinson, (1996) "A note on reliable full-duplex transmission over half-duplex links *Commun. ACM*, (**12**): 260-261.

Bergstra, J. A. and J. W. Klop, (1986) Verification of an alternating bit protocol by means of process algebra In *Proceedings of the International Spring School on Mathematical method of specification and synthesis of software systems '85*, , New York, NY, USA. Springer-verlag New York, Inc. 09–23.

Behrmann, G., A. David, and K.G. Larsen. (2004) A tutorial on UPPAAL. In *Proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFMRT '04)*, number 3185 in LNCS, Springer-Verlag. 200-236.

Chapter Finite state description of communication protocols, (1995). IEEE Computer Society Press, Los Alamitos, CA, USA, 66-77.

Cleavel R. (1993) Analysing Concurrent Systems Using the Concurrency Workbench. In *Proceeding of the Functional Programming, Concurrency, Simulation and Automated Reasoning: International Lecture Series 1991-1992, McMaster University, Hamilton, Ontario, Canada,* London, UK. Springer-Verlag. 129–144.

Cleaveland, R., J. Parrow, and B. Steffen. (1993) The concurrency workbench: a semantics-based tool for the verification of concurrent systems. *ACM Trans. Program. Lang. Syst.*, (**15**): 36–72.

Gregor. V. B. Conformance testing methodologies and architectures for OSI protocols.

Holzmann, G. J. (1995) “Design and validation of computer protocols,” Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Lynch. W. C. (1998) Computer Systems: Reliable full-duplex file transmission over half-duplex telephone line. ACM, (**11**): 407–410.

Yemini Y. and J. F. Kurose. (1982) Can current protocol verification techniques guarantee correctness? *Computer Networks*, 6 (**6**): 377–381.

Yehuda A., H. Attiya, A. Fekete, M. Fischer, N. Lynch, Y. Mansour, Dai-Wei. Wang, and L. Zuck. (1994) Reliable Communication Over Unreliable Channels. *J. ACM*, 41 (**6**): 1267–1297.