Model Checking Dependability Attributes of Wireless Group Communication — Full version

Mieke Massink
C.N.R.-ISTI, Via Moruzzi 1,
I-56124 Pisa, Italy,
M.Massink@isti.cnr.it

Joost-Pieter Katoen
Dept. of Computer Science,
University of Twente,
P.O. Box 217, 7500 AE Enschede,
The Netherlands
katoen@cs.utwente.nl

Diego Latella
C.N.R.-ISTI, Via Moruzzi 1,
I-56124 Pisa, Italy,
D.Latella@isti.cnr.it

Abstract

Models used for the analysis of dependability and performance attributes of communication protocols often abstract considerably from the details of the actual protocol. These models often consist of concurrent sub-models and this may make it hard to judge whether their behaviour is faithfully reflecting the protocol. In this paper, we show how model checking of continuous-time Markov chains, generated from high-level specifications, facilitates the analysis of both correctness and dependability attributes. We illustrate this by revisiting a performance analysis [9] of a variant of the central access protocol of the IEEE 802.11 standard for wireless local area networks. This variant has been developed to support real-time group communication between autonomous mobile stations. Correctness and dependability properties are formally characterised using Continuous Stochastic Logic and are automatically verified by the ETMCC model checker. The models used are specified as Stochastic Activity Nets.

1. Introduction

Continuous-time Markov chains (CTMCs) are widely used to analyse important system dependability and performance issues. Usually such CTMCs are not obtained in a direct way, due to the increasing complexity of the analysed systems, but are generated automatically from higher-level specifications such as e.g. Stochastic Activity Nets (SAN) [22] for which mature tool-support is available. Such tools support both the development of high-level specifications and the calculation of relevant measures, e.g. steady-state and transient state probabilities.

The high-level specifications used for the analysis of dependability and performance aspects of communication protocols often abstract considerably from the details of the actual protocol. In many cases, these models, in their turn, consist of concurrently composed sub-models. This may complicate the judgement whether their behaviour is faithfully reflecting the protocol. The extension of dependability tools with model-checking capabilities and a temporal logic allows for the verification of behavioural aspects as well as for the convenient, concise and unambiguous specification and automated verification of dependability and performance measures.

In this paper, we illustrate these advantages in practice by revisiting part of a dependability and performance analysis of a variant of the *centralised* medium access protocol of the IEEE 802.11 standard for wireless local area networks [4, 9, 10]. This variant has been developed to provide reliable real-time group communication within teams of autonomous mobile robotic systems over wireless (radio) networks [23, 20]. In the IEEE 802.11 standard, the problem of message loss is addressed by defining two alternating periods of medium access control; a centralised one suitable for the exchange of time-critical messages, and a distributed one, suitable for less time critical or non-time critical messages. The *distributed* medium access control mechanism for non time-critical communication over wireless networks has been studied in [17] applying probabilistic model checking techniques.

Group communication between autonomous mobile stations via wireless local area networks presents particular problems due to the *locomotion* of the mobile stations and the *unshieldedness* of wireless communication. It is therefore susceptible to a high degree of message losses in a bursty fashion.

The variant of the protocol that we consider in this paper proposes to reduce the number of retransmissions required to guarantee reliable communication in order to improve the real-time performance of the protocol. The reduction of reliability due to fewer retransmissions is compensated for by a mechanism of active acknowledgments and the distribution of decision information that is included in the header of broadcasted messages. For many applications on real-time mobile stations the reduced reliability does not cause a serious problem as long as all mobile stations in the network agree in time not to deliver a message to their application when there is some station that did not receive the user message, viz. the property of agreement is satisfied.

In this paper, we analyse the models developed in [4, 9, 10] to determine the probability that a station misses a decision message and the probability that a user message is never delivered. First we check the correctness of the analytic model by generating the CTMC using the UltraSAN tool [22] and verify correctness properties of the concurrent model by the prototype stochastic model checker ETMCC (Erlangen Twente Markov Chain Checker) [14]. This model checker allows for the verification of both qualitative and quantitative (stochastic time) properties expressed as formulas of the (stochastic) branching time logic CSL (Continuous Stochastic Logic) [1, 2]. UltraSAN is a software package for model-based evaluation of systems represented as SAN's. It provides analytic solvers as well as discrete-event simulators but it has no model-checking facilities.

The contribution of this paper is threefold. We use a model checking approach on a case study on which numerical and experimental results are available in the literature. We show that the model checking capability to verify both qualitative and quantitative properties of concurrent models can greatly enhance the effectiveness of existing dependability analysis tools to increase the confidence in the accuracy and faithfullness of the models on which the analyses are based. In fact, its automatic verification reveals serious problems of the existing model and gives rise to the development of a *new* model that more faithfully reflects the synchronous broadcast aspects of the protocol. We show this by comparing the verification results for qualitative and quantitative properties for both models. Finally, the direct link between the high-level specification in SAN and the derived CTMCs on which the analysis by both UltraSAN and ETMCC are based gives an opportunity to compare the results and to obtain feedback on the performance of the tools.

The paper is organised as follows. Section 2 introduces the Markov chain model checker ETMCC and the stochastic temporal logic CSL. Section 3 briefly describes the relevant aspects of the group communication protocol proposed in [23]. In Section 4, we discuss the SAN models that form the basis for the verification. Section 5 is dedicated to the formalisation and verification of qualitative correctness properties and quantitative dependability measures. Section 6 presents a more faithful model for the evaluation of decision failures, and we compare the models and the outcomes produced by UltraSAN and the ETMCC model checker. Finally, in Section 7 we discuss the main conclusions and outline future research.

2. Model Checking Dependability

In the model checking approach to dependability analysis a *model* of the system under consideration is required together with a desired *property* or *dependability measure*. Model checking provides a systematic check whether the given model satisfies the property. Effective, optimised model checking algorithms have been developed to dramatically reduce the state space that needs to be searched, and to keep its representation compact as well using sophisticated techniques such as Binary Decision Diagrams [7].

Typically, models are finite-state automata, where transitions model the evolution of the system while moving from one state to another. These automata are usually generated from a high-level description language. In the case of stochastic modelling, such models are typically CTMCs and languages such as stochastic Petri nets, stochastic process algebras or SANs are used to generate them.

In the model checking approach, the properties are usually expressed in some form of temporal logic. In this paper the Continuous Stochastic Logic [1, 2] is used, which is a stochastic variant of the well-known Computational Tree Logic (CTL) [8]. CTL allows for stating properties over *states*, and over *paths* using the following syntax:

State-formulas

$$\Phi ::= a \;\Big|\; \neg\,\Phi \;\Big|\; \Phi \,\vee\, \Phi \;\Big|\; \exists\varphi \;\Big|\; \forall\varphi$$

a	: atomic proposition
$\exists\varphi$	: there *Exists* a path that fulfils φ
$\forall\varphi$	: *All* paths fulfil φ

Path-formulas

$$\varphi ::= X\,\Phi \;\Big|\; \Phi\,\mathcal{U}\,\Phi$$

$X\,\Phi$	: the *neXt* state fulfils Φ
$\Phi\,\mathcal{U}\,\Psi$	: Φ holds along the path, *Until* Ψ holds
$\Diamond\Phi$	: $\mathsf{true}\,\mathcal{U}\,\Phi$, i.e., eventually Φ
$\Box\Phi$	: $\neg\Diamond\neg\Phi$, i.e., invariantly Φ

The meaning of atomic propositions (a), negation ($\neg$) and disjunction ($\vee$) is standard; note that using these operators, other boolean operators such as conjunction ($\wedge$), implication ($\Rightarrow$), true (true) and false (false), and so forth, can be defined. The state-formula $\exists\varphi$ is valid in state s if there *exists* some path starting in s and satisfying φ. The formula $\exists\Diamond\mathit{deadlock}$, for example, expresses that for some system run eventually a deadlock can be reached (potential deadlock). On the contrary, $\forall\varphi$ is valid if *all* paths satisfy φ; $\forall\Diamond\mathit{deadlock}$ thus means that a deadlock is inevitable. A path satisfies an until-formula $\Phi\,\mathcal{U}\,\Psi$ if there is a state on the path where Ψ holds, and at every preceding state on the path, if any, Φ holds.

To specify and evaluate performance and dependability measures as logical formulas over CTMCs, CSL [1, 2] has been developed. CSL extends CTL by two probabilistic operators that refer to the steady-state and transient behaviour of the system being studied. Whereas the steady-state operator refers to the probability of residing in a particular set of *states* (specified by a state-formula) in the long run, the transient operator allows us to refer to the probability of the occurrence of particular *paths* in the CTMC. In order to express the time-span of a certain path, the path-operators until $\mathcal{U}$ and next X are extended with a parameter that specifies a time-interval. Let I be an interval on the real line, p a probability value and $\bowtie$ a comparison operator, i.e., $\bowtie\, \in \{<, \leqslant, \geqslant, >\}$. The syntax of CSL is:

State-formulas

$$\Phi ::= a \;\Big|\; \neg\,\Phi \;\Big|\; \Phi \,\vee\, \Phi \;\Big|\; \mathcal{S}_{\bowtie p}(\Phi) \;\Big|\; \mathcal{P}_{\bowtie p}(\varphi)$$

$\mathcal{S}_{\bowtie p}(\Phi)$	: prob. that Φ holds in steady state $\bowtie p$
$\mathcal{P}_{\bowtie p}(\varphi)$	: prob. that a path fulfils $\varphi \bowtie p$

Path-formulas

$$\varphi ::= X^I\,\Phi \;\Big|\; \Phi\,\mathcal{U}^I\,\Phi$$

$X^I\,\Phi$	: next state is reached at time $t \in I$ and fulfils Φ
$\Phi\,\mathcal{U}^I\,\Psi$	: Φ holds along path until Ψ holds at $t \in I$

The state-formula $\mathcal{S}_{\bowtie p}(\Phi)$ asserts that the steady-state probability for the set of Φ-states meets the bound $\bowtie\, p$. The operator $\mathcal{P}_{\bowtie p}(.)$ replaces the usual CTL path quantifiers $\exists$ and $\forall$. In fact, $\exists\varphi$ can be written as $\mathcal{P}_{>0}(\varphi)$ and $\forall\varphi$ as $\mathcal{P}_{\geqslant 1}(\varphi)$. This allows for the expression of qualitative as well as stochastic properties in CSL. We shall frequently use this aspect in this paper. $\mathcal{P}_{\bowtie p}(\varphi)$ asserts that the probability measure of the set of paths satisfying φ meets the bound $\bowtie\, p$. Temporal operators like $\Diamond$, $\Box$ and their real-time variants $\Diamond^I$ or $\Box^I$ can be derived, e.g., $\mathcal{P}_{\bowtie p}(\Diamond^I\,\Phi) = \mathcal{P}_{\bowtie p}(\mathsf{true}\ \mathcal{U}^I\,\Phi)$ and $\mathcal{P}_{\geqslant p}(\Box^I\,\Phi) = \mathcal{P}_{<1-p}(\Diamond^I\,\neg\Phi)$. The untimed next- and until-operators are obtained by $X\Phi = X^I\Phi$ and $\Phi_1\,\mathcal{U}\,\Phi_2 = \Phi_1\,\mathcal{U}^I\,\Phi_2$ for $I = [0, \infty)$. Of course, also in CSL other boolean operators can be defined in the same way as in CTL.

CSL allows for the expression of four different types of performance and dependability measures, viz. steady-state measures, transient-state measures, path-based measures, and nested measures. In this paper we shall use several transient-state measures and nested measures.

The ETMCC model checker [14] is a prototype tool that supports the verification of CSL-properties over CTMCs. It checks the validity of CSL-properties for all states in the model and provides feedback on the calculated probabilities for *each* state where appropriate. The model checker takes as input a model file with a textual representation of a CTMC, a label file associating each state to the atomic propositions that hold in that state and a given accuracy. ETMCC is based on sparse matrix representations of CTMCs. Alternative model checkers for CSL include PRISM [16], Prover [24] and the APNN (Abstract Petri Net Notation) toolbox [5]. PRISM uses symbolic data structures (i.e., variants of Binary Decision Diagrams) whereas Prover is based on discrete event simulation, rather than numerical computation.

3. Group Communication Protocols for Wireless Local Area Networks

Real-time group communication protocols for wireless local area networks are very important for applications where autonomous mobile stations are expected to cooperate and synchronise their behaviour in order to accomplish a common goal. Prominent examples of (higher level) protocols that make use of such communication protocols are those that coordinate cooperation at a higher level of abstraction, such as: consensus protocols, leader election protocols, protocols that guarantee the consistency of replicas and non-blocking atomic transactions.

A real-time group communication protocol needs to:

- guarantee real-time communication, i.e., it needs to guarantee an upper bound on the delay of message communication,
- provide reliable communication,

- be able to handle failure of mobile stations in a group and keep the stations informed about the status of each station, and finally,

- guarantee that all stations receive the same messages in the same order.

The last requirement means that the protocol is supposed to implement a kind of distributed 'Blackboard'. On the Blackboard all present mobile stations can share their information (reading and writing) and the status of each station (being alive/reachable or unreachable).

The main problem that a real-time group communication protocol for wireless networks needs to overcome is the high degree of message losses. This high degree of losses is caused by the unshieldedness of the wireless medium, and partially also by the velocity of the mobile stations. A further characteristic of these losses is their bursty nature, which means that often series of consecutive messages are lost.

In the IEEE 802.11 standard [15], the problem of message losses and the real-time communication requirement have been addressed by the introduction of two alternating periods of medium access control. In the Contention Period (CP), distributed medium arbitration takes place and collisions may occur. The arbitration scheme used during CP is standard carrier sense multiple access with collision avoidance (CSMA/CA). This period is useful for the exchange of non time-critical or less time-critical messages. The Contention Free Period (CFP) has a centralised medium arbitration and the group members get exclusive access right to communicate over the shared medium. The CFP is specifically designed for real-time communication. The two periods, CP and CFP, are activated in an alternating way under the control of a central Access Point (AP). This is a special fixed node in the network with a central position with respect to the mobile stations with which it communicates to obtain an optimal reachability. Both periods can have variable length which is decided by the AP and which can be used to optimise the communication.

During the CFP the AP coordinates the medium access for all stations that are reachable over the wireless network (see Fig. 1). At the beginning of the CFP all stations remain silent, except for the AP that transmits a polling message to some station in the group. When a station receives a polling message it may broadcast a message over the network. The polling strategy is decided by the AP which is also in charge of assigning a sequence number to the broadcasted message in order to make total ordering of messages possible.

The real-time group communication protocol that we analyse in this paper is a variant of the protocol used for the CFP in the IEEE 802.11 standard and has been developed by Mock et al. [20].

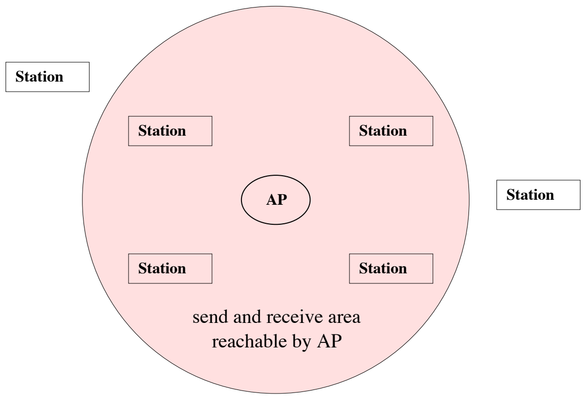


Figure 1. Centralised medium arbitration in the IEEE 802.11 standard

3.1. Basic operation of the real-time group communication protocol

The protocol is based on a *dynamic redundancy scheme*. In this scheme a message is only retransmitted upon the detection of its loss. Such a scheme needs the explicit recognition of communication failures and an acknowledgment strategy that reports the status of a transmission.

The protocol is based on the following fault assumptions about the wireless network [9]:

- if a message is delivered (during the CFP), it is delivered correctly and within a fixed time bound (t_m).

- messages may be lost, possibly in an asymmetric way, i.e. some stations may receive a broadcast message while others do not. In any case, the number of consecutive losses of the *same* message is assumed to be bounded by the so-called *omission degree* OD.

- the AP is reliable, i.e. it is not subject to any kind of error.

- stations may suffer from crash failures or leave the reach of the AP.

The group communication protocol is structured into *rounds* and it is assumed that there is a maximum number n_{max} of stations in a group and that $N \leqslant n_{max}$ is the current size of the group. A round is composed of a series of slots, one for each station in the group, where each slot consists of a triple of message transmissions; a *polling* message from the AP to the station, a *broadcast request* message from the station to the AP and a *broadcast* message by which the AP sends the user message of the sending station to all stations. Each round is identified by a unique round number, starting from 0 and incremented by 1 at the beginning of each new round. The AP polls each station of the group exactly once in each round, and polls the stations always in the same order, sending them the round number in the polling message. After being polled, station s (denoted as originator in the following) sends a broadcast request message to the AP. This message is composed of an acknowledgment field, a local sequence number and a *user* message m. The acknowledgment field is composed of N bits, numbered from 0 to $N-1$ (from left to right), each of which is used to acknowledge one of the N preceding broadcasts. The bit at position 0 encodes the acknowledgment for the $N-1$th but last broadcast message that the AP has sent in the last round after polling station s. The bit at position $N-1$ indicates this for the last broadcast message. This acknowledgment scheme implies that exactly one round after broadcasting a user message of a certain station, the AP is able to decide whether each group member has received the user message or not. Fig. 2 shows an illustration of the acknowledgment scheme (see also [23]).

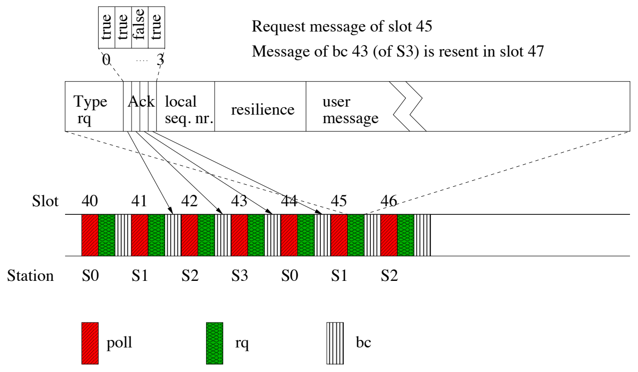


Figure 2. Acknowledgment scheme in broadcast request messages

When station s is polled three cases are distinguished. In the first case, all group members, including the originator (and currently polled station) s, have received the user message m sent by s in the previous round. This means that s can send its next user message m' (if available) to the AP as response to the polling message. In the second case, not all group members have received user message m, but the originator (and currently polled station) s *has* received it. This last fact serves as an implicit acknowledgment for s that the AP has received user message m (i.e. nothing went wrong with the broadcast request message containing m from s to the AP). In this case, the AP will retransmit user message m from s which the AP keeps in its memory, and it will accept the new user message m' from s (if available). This new user message will be stored until the current user message m has been successfully transmitted to all stations in the group. In the third case, not all group members have received user message m, and in particular the originator (and currently polled station) s has not received it. In this case s cannot be sure that the AP has received the user message in the first place (i.e. something may have gone wrong with the broadcast request message containing m from s to the AP), and it therefore retransmits user message m to the AP. Note that in this way the AP stores at most one extra user message from every station.

Summarising, station s has three possibilities for the data field m of the broadcast request message:

- m contains the previous user message if s has not received the broadcast of its last request containing that user message,
- m contains the current user message, if s has received the broadcast of its last request containing the previous user message, or
- m is empty, if s has no new user message to send and if s has received the broadcast of its last request containing the previous user message.

When the AP receives the broadcast request message containing user message m from s there are two possibilities. Either, the AP still has the previous user message of s that has not yet been acknowledged by all stations and which thus has to be retransmitted by the AP, or the previous user message has been successfully acknowledged by all stations. In the first case, the AP stores the new user message for later transmission when the previous user message of s has been acknowledged by all stations in the group. In the second case, the AP assigns a global sequence number (sg) to m, broadcasts it and increments the sequence number by 1. The broadcast message is composed of sg, a local (to the sending station) message sequence number (sl) and the user message m. The identity of the original sender of the user message m may be reconstructed by the receiving station by means of the global sequence number. E.g. if no station fails, then the identity is equal to $sg \ mod \ N$.

When a station receives a request from the user to broadcast a user message, it immediately sets a variable to the number of the last round in which that user message can be sent to the AP. This can be determined because each user message can be sent at most $OD + 1$ times and is then assumed to have arrived at least once at all stations in the group (see fault assumptions before).

After polling a station s, the AP expects to receive a broadcast request message from s within $2 * t_m$ time units. If it does not receive this request message, it considers the request message (or the polling message) to be lost. It then retransmits the previous user message of s, with all bits of the acknowledgment field of the broadcast message set to false, if not all stations have acknowledged the user message yet. Otherwise, it sends a broadcast message with an empty user message field. Subsequently, the AP polls the next station. If the AP has not received the request message from s for more than OD times (i.e. after OD rounds have passed), it considers s to have left the group. In the header of the broadcast message that follows the polling of s it indicates the change in group membership. Reintegrations and associations of new group members are handled in the non real-time phase (CP).

3.2. Further enhancement of the protocol

The improvement of the protocol proposed in [23] attempts to achieve a further decrease in the latency of real-time messages by reducing the maximum number of message retransmissions from OD to a user-specified number lower than OD. This number is the so-called *resilience degree* (res). With this reduction of the number of retries full reliability of the protocol can no longer be guaranteed under the assumptions about the network (as long as $res < OD$). This means that it may happen that a user message that is broadcasted is not received by all stations in the group within res retransmissions. This is not a serious problem for many applications as long as all stations agree in time not to deliver that message to their respective application processes.

Thus, the stations are allowed to *deliver* a user message to the application only if the message is received by *all* stations. This is decided by the AP, based on a positive acknowledgment for that user message from every station. The decision of the AP is to be communicated in a reliable and timely way. This is achieved by means of the transmission of the *decision* for each user message through a field in the header of each broadcast message composed of $OD + 1$ bit-*pairs*. Every decision is retransmitted $OD + 1$ times, so there is no need for an acknowledgment of the reception of the decision by the station (given the network assumptions). For each message broadcasted, the AP takes one of the following decisions:

- a *positive* decision if the previous user message has been transmitted $OD + 1$ times or when the AP received a positive acknowledgment for the user message from each station. This is encoded by (true,true).

- a *negative* decision, when at least one station has not given a positive acknowledgment for a user message after res number of re-transmissions. This is encoded by (true, false).

- a *don't know* decision if not all stations have given a positive acknowledgment, but the number of retransmissions is still less than res. This is encoded by (false,false).

(The fourth bit-pair (false,true) is used to communicate that a station has left the group. We will deal with this later on.) The AP uses the $OD + 1$ bit-pairs decision field, numbered from 0 to OD (from left to right), as a FIFO channel, in the sense that every time it sends a broadcast message, it shifts all bit-pairs one place to the right, so to the positions 2 to OD and the last bit-pair gets kicked-out. It then inserts the decision related to the user message in the current broadcast in position 0 — i.e. the left-most — of the decision field. The rotation of this bit-pair field guarantees that each decision is broadcasted exactly $OD + 1$ times.

Each station needs to interpret the information in the bit-pair field of the broadcast message. If it has not lost a broadcast message between the current and the previous one received, it only needs to inspect the most recent bit-pair at position 0. A station can know whether it missed a broadcast by checking the global sequence number sg. The decision in position 0 relates to the current message, so the station knows what to do with that message (delivering or not or not yet).

If a station lost some broadcasts, it can use the decision field to reconstruct the situation. It only has to inspect those bit-pairs that relate to the lost broadcasts and deal with them, starting from the oldest decision pair up to the one in position 0. So, suppose the last broadcast message that the station s had received was $sg1$ and that the current broadcast message has number $sg2$; then it needs to start processing the bit-pair at position $sg2 - sg1$ and work backward to position number 0. Note again that a station can never loose more than OD consecutive broadcast messages.

From the current sg and the position of the bit-pairs, the receiving station can also reconstruct the sg belonging to the decisions and thus knows the slot in which the user message was sent and consequently the identity of the station that sent the user message. If the decision is positive, the receiving station *must* have received the current user message of that sending station, and can therefore deliver it to the application.

If the decision is negative, or a 'don't know', it may be that the station has not yet received the user message. In the case of a negative decision it does not care, in case of a 'don't know' the station continues to wait for the user message. Fig. 3 shows the information in a broadcast message for a typical situation (adapted from Schemmer [23]).

If the AP has been polling a station $OD + 1$ times (so, in $OD + 1$ consecutive rounds), and has not received a broadcast request message from that station as a reaction to the polling message, the AP assumes that the station crashed, or has become unreachable and considers it to have left the group. This change in the group composition is communicated to the other stations by the AP using the bit-pair combination (false,true), which is inserted in the decision field of the broadcast messages like any other decision message.

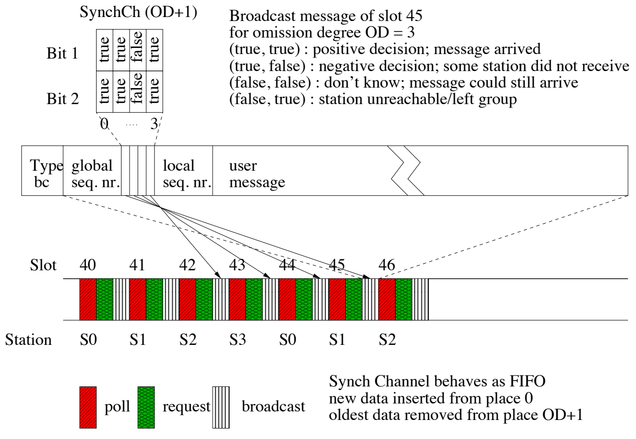

SynchCh (OD+1)
Broadcast message of slot 45
for omission degree OD = 3
(true, true) : positive decision; message arrived
(true, false) : negative decision; some station did not receive
(false, false) : don't know; message could still arrive
(false, true) : station unreachable/left group
Bit 1
Bit 2
true
true
false
true
true
true
false
true
0
3
Type bc
global seq. nr.
local seq. nr.
user message
Slot
40
41
42
43
44
45
46
Station
S0
S1
S2
S3
S0
S1
S2
poll
request
broadcast
Synch Channel behaves as FIFO
new data inserted from place 0
oldest data removed from place OD+1

Figure 3. Broadcast message and decision information

This guarantees that all remaining stations know at most $OD + 1$ broadcasts later that a station has left the group and also know its identity. When a station discovers that another station s' has left the group, it communicates this to the user and updates its information about which stations are members of the group. It does so by decrementing the identity of each station with an identity higher than that of s' by one. The AP does not poll the excluded station s' anymore (within the same CFP), but instead polls the next station in the slot of s'. The stations update the length of the acknowledgment field to the new group size.

3.3. Correctness properties

A protocol that satisfies the properties of validity, agreement and integrity, as defined below and in [9], is considered to be reliable. In the description of these properties below a 'correct' station indicates a station that has not crashed and that is a member of the group.

validity: A message broadcasted by a correct station is eventually delivered (i.e. forwarded by the station to the application process on the receiving site) by every correct station, as long as the number of message losses it incurs is less than a specified value.

agreement: A message delivered by *one* correct station is eventually delivered by *every* correct station.

integrity: For any message m, every correct station delivers m at most once and only does so if m has been broadcasted.

total order: If the messages m_1 and m_2 are delivered by correct stations s_1 and s_2, then station s_1 delivers message m_1 before message m_2 if and only if station s_2 delivers message m_1 before m_2.

timeliness: A message broadcasted by a correct station at time t is either delivered to all correct stations before time $t + \Delta t$, or not at all, for some known positive constant Δt.

The functional correctness of the group communication protocol has been treated extensively in [23] where also a specification of the protocol is given in SDL (Specification and Description Language [19].) An analysis of the real-time performance of the protocol is provided in [20]. All above listed properties are proven to hold for this variant of the protocol, with the exception of the validity property, which is not strictly necessary for a significant class of applications, as long as the other properties are satisfied, in particular the agreement property. Neither of these works have used automatic verification tools for the verification of the properties.

3.4. Dependability measures

A number of dependability and performance measures for the protocols are addressed in [9], where a numerical analysis has been carried out by means of the UltraSAN tool [22]. We revisit two of these in this paper, but following a model-checking approach for their analysis. We start from the high-level SAN specifications in [9] and formulate the measures as CSL formulas.

The two dependability measures that we address are:

$P_{D>OD}$**:** The probability that a *decision message* (i.e. a message issued by the AP to commit or abort the delivery of a broadcast message) is not received by at least one station in the group, within the interval of time T_{CFP} (duration of the CFP phase). This measure represents an estimate of the probability for the protocol to fail in an undetected and undesirable way with possible catastrophic consequences on the system and its users. Therefore, this probability should be sufficiently low.

P_{UM}**:** The probability that the AP does not receive acknowledgments for a user message by all the stations within res retransmissions within T_{CFP}. In this case, the AP broadcasts to all stations in the group the decision not to deliver that message to their applications. In other words, P_{UM} is the probability that some station in the group has not acknowledged a user message sent by the AP after res retransmissions. This property gives an indication to which extent the validity property is violated.

4. A Dependability Model

In [9], a model is developed that covers relevant aspects of the protocol and its environment that are necessary to analyse the dependability measures of interest. A single model is used to analyse several dependability measures by varying the values of its parameters.

4.1. Fading model

In modelling the environment, the interference between different versions of the transmitted signal and the Doppler shift caused by the relative motion of receiving and sending stations, has been taken into account. Both effects cause the so-called *signal fading phenomenon*. The probability of message loss resulting from fading signals has been approximated by the first-order discrete time Markov chain (DTMC) [25] depicted in Fig. 4. The DTMC has two states, S and F, standing for (previous) success and failure of a communication respectively. If the previous communication has been successful, with probability p the next communication will also be successful. With probability $1 - p$, the next communication will be a failure. If the previous communication has failed, then with probability q the next communication fails, and with probability $1-q$ it is successful. This behaviour can be presented as a transition probability matrix in a standard way. The probability of success or failure of a number of consecutive message losses (success) can be obtained by matrix multiplication. The parameters p and q have been derived considering the communication between the AP and the stations as Rayleigh fading channels and using experimental data available to calculate the approximate values [9]. In particular, p and q are functions of the steady state probability that a communication fails (PE) and the normalized Doppler frequency. For details we refer to [9, 25].

$$\begin{tikzcd} S \arrow[loop left, "p"] \arrow[r, bend left, "1-p"] & F \arrow[loop right, "q"] \arrow[l, bend left, "1-q"] \end{tikzcd}$$

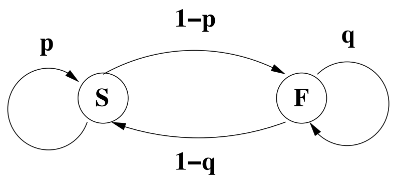


Figure 4. DTMC modelling channel fading

4.2. Station model

The fading model has been integrated into the model of a station defined using the SAN formalism [18] which is shown in Fig. 5. SANs are a high-level modelling formalism for the specification of dependability and performability models. SAN models consist of four primitive objects: places, activities, input gates and output gates. *Places* represent the state of the system and are marked by tokens, like in Petri nets. *Activities* represent transitions or actions of the system. *Input gates* are used to control the enabling of activities, and *output gates* are used to change the state of the model on completion of an activity. The model parameters can be defined as global variables of type double ($GLOBAL\_D$) or short ($GLOBAL\_S$). The definition of the objects used in Fig. 5 are given in Table 1.

Let us briefly explain the SAN model. The place $POLL$ models that the station is polled by the AP. Initially it has one token. The input gate chk enables the activity $nprb$ only if there is a token in place $POLL$ and no token at place $FAIL$. If this condition is satisfied, chk removes the token from place $POLL$. chk models that a station only reacts to a polling message from the AP if the station did not fail. The exact interpretation that is given to the failing of a station depends on the dependability measure that is analysed. For the analysis of $P_{D>OD}$ a station fails if it missed more than OD consecutive decision messages. In the case of P_{UM} it fails if it missed more than res consecutive user messages. The timed activity $nprb$ (probabilistic broadcast) models the performance aspects of the wireless network and forms the central part of the model.

The SAN model can be used for the analysis of several dependability measures by selecting proper values for the probabilities and the rate of the timed activity.

Model for the analysis of $P_{D>OD}$. When the model is used to analyse property $P_{D>OD}$, the time distribution function is chosen to be exponential with a rate being the reciprocal of the duration of one slot, i.e. the sum of the transition time of one polling message, a broadcast request message and a broadcast message. The polling message is very short (app. 100 bytes), but the broadcast messages are longer (app. 1000 bytes). Let TP be the *mean time* required for the polling message to be transferred from the AP to a station, and TM the same for a broadcast message. Then the exponential distribution rate of a slot is:

$$1/(2 * TM + TP)$$

The timed activity $nprb$ has two cases, represented as two small circles attached to the hollow oval in Fig. 5. The probability distribution of the two cases is defined by the case distribution and may also depend on the marking of the network at the moment of completion of the activity. In this model, the distribution depends on the marking of place $SUCCESS$. A token in place $SUCCESS$ means that the previous triple of polling, broadcast request and broadcast messages, has been a success. We obtain the fading characteristics as the outcome of the product of three matrices $M'.M.M$. Here M' represents the matrix defining the fading characteristics of the short polling message, with its characteristic probabilities p' and q', and M defines the fading characteristics of a broadcast message with its respective probabilities. Let P and Q be the resulting probabilities of this matrix multiplication, i.e. P is the probability in that resulting matrix of the self-loop from state S to itself, $1 - P$ is the probability in that matrix of the transition from S to F, etc. The probabilities associated with the two cases in the timed activity $nprb$ are then derived from the DTMC in Fig. 4 where p and q are now P and Q. So case 1, denoting a broadcast failure, connected to output gate $FAIL\_BC$ becomes $1-P$ and case 2, denoting a successful broadcast, becomes P. If there is no token on place $SUCCESS$, the probabilities for the two cases are Q and $1 - Q$, respectively.

The output gate $FAIL\_BC$ removes any token from place $SUCCESS$, increments the number of tokens on place $COUNTER$ by one, and if the number of tokens on $COUNTER$ exceeds the omission degree OD, it puts a token on place $FAIL$. Otherwise, it puts a token on place $POLL$, modelling that the station is ready for the next communication (triple). The $COUNTER$ represents the number of consecutive failed communications for a given station.

The output gate $SUCC\_BC$ changes the state of the model after a successful broadcast has taken place. It puts a token on place $SUCCESS$, resets $COUNTER$ to zero (i.e. removes all its tokens) and puts a token on place $POLL$.

Initially, there is one token on place $SUCCESS$ and on place $POLL$, and all other places are empty.

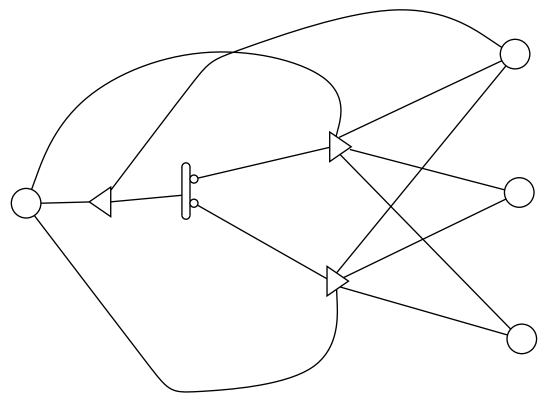


Figure 5. UltraSAN model for one station

The UltraSAN tool provides a mechanism for replicating a single station. This allows for the easy generation of a model with N stations that may share places. In this case the stations share place $FAIL$.

Model for the analysis of P_{UM}. The model used for

Timed activity	nprb
Exp. distr. rate	1/(2*GLOBAL_D(TM) + GLOBAL_D(TP))
Case 1	if (MARK(SUCCESS)==1) return (1-GLOBAL_D(P)); else return GLOBAL_D(Q);
Case 2	if (MARK(SUCCESS)==1 return (GLOBAL_D(P)); else return (1-GLOBAL_D(Q));
Output gate	FAIL_BC
Output function	MARK(SUCCESS)=0; MARK(COUNTER)++; if (MARK(COUNTER) > GLOBAL_S(OD)) {MARK(FAIL)=1; } else MARK(POLL)=1;
Output gate	SUCC_BC
Output function	MARK(SUCCESS)=1; MARK(COUNTER)=0; MARK(POLL)=1;
Input gate	chk
Input predicate	(MARK(FAIL)==0) && (MARK(POLL)==1)
Input function	MARK(POLL)=0
Initial marking	
FAIL	0
COUNTER	0
SUCCESS	1
POLL	1

Table 1. Specification details of SAN model elements

analysing property P_{UM} is the same as that for $P_{D>OD}$ except for the values of P and Q and the rate of the exponential distribution of the timed activity. In fact, for P_{UM} we are interested in the probability that a *user message* is not received by a station within res retransmissions and within the duration of the CFP. This means that we need to set the rate of the timed activity to the reciprocal of the mean duration of one round, i.e. to $1/(N * (2 * TM + TP))$ where N is the number of stations in the group, which is equal to the number of slots in a round, and TM and TP stand for the mean duration of the transmission of a broadcast and a polling message respectively. The probabilities P and Q have now to be based on a round as well. They can be obtained as the result of the matrix multiplication $[M'.M.M]^N$ in the same way as for the model for $P_{D>OD}$.

5. Model Checking Properties

5.1. CTMC model of a station

The one-station SAN model has a very simple underlying Continuous Time Markov Chain (CTMC) that can be obtained automatically using UltraSAN. In fact, UltraSAN generates an optimized CTMC which is called the reduced base model (RBM). This model is an internal representation that UltraSAN uses for numerical analysis [21]. Fig. 6 shows a graphical representation of the underlying CTMC that is parametric in OD, where $T = TP + 2 * TM$. This CTMC has $OD + 2$ states $M1$ through $M(OD + 2)$. State $M1$ corresponds to the initial state of the SAN model, where a station is polled and the previous communication has been a success. With rate $(1 - P)/T$, a communication (triple) fails and state $M2$ is reached, representing the counter being equal to one. With rate P/T the communication is successful, and we remain[1] in state $M1$. From $M2$, we can either go back to $M1$ with rate $(1-Q)/T$ in case of a successful communication (note that the probability used is now Q because in state $M2$ the previous communication was a failure) or move to $M3$ with rate Q/T in case of an unsuccessful communication. In state $M(OD + 1)$ the counter has the value OD. So, one more unsuccessful communication (with rate Q/T) brings us to state $M(OD + 2)$, where the counter reaches the value $OD + 1$, corresponding to having a token in the $FAIL$ place of the SAN model. This is an absorbing state, modelling that after the station fails, it can no longer communicate with the AP. From state $M3$ through state $M(OD + 1)$, with rate $(1 - Q)/T$ a successful communication can take place, which brings us back to the initial state $M1$.

[1]Note that the CTMCs generated by UltraSAN may contain self-loops.

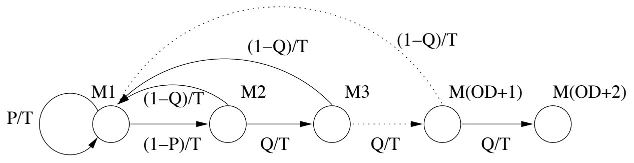


Figure 6. CTMC for one station which is parametric in OD

Ap	Meaning	M1	M2	M3	M4
$@poll$	token on POLL	1	1	1	0
$@succ$	token on SUCCESS	1	0	0	0
$@fail$	token on FAIL	0	0	0	1
$c = 0$	COUNTER = 0	1	0	0	0
$c = 1$	COUNTER = 1	0	1	0	0
$c = 2$	COUNTER = 2	0	0	1	0
$c = 3$	COUNTER = 3	0	0	0	1

Table 2. Atomic propositions for CTMC for single station and $OD = 2$

5.2. Generating input for ETMCC

The format of the model file generated by UltraSAN is different from that required by the ETMCC model checker [14], but contains all the information needed to construct the proper model file for ETMCC. We developed a simple program to perform this transformation.

The association between the markings of the SAN model and the CTMC facilitates setting up a proper labelling file that is also part of the input for the ETMCC model checker. The labelling file defines the atomic propositions that hold in the various states of the CTMC. Table 2 shows the atomic propositions (Ap) that correspond to the markings of the SAN model for one station with $OD = 2$ and indicates in which states of the CTMC they hold.

A simple program has been developed to transform the marking file generated by UltraSAN into a proper label-file for ETMCC. The atomic propositions are used to state interesting properties of the model in a precise and formal way using CSL.

5.3. Properties for single-station model

In order to better understand the behavioural aspects of the model that is used for the analysis, we proceed by showing a number of qualitative and quantitative properties and discuss the result of their verification. We first address properties of the very simple single-station model, and then

N	1	TM	7.646 ms
OD	2	TP	2.380 ms
P	0.999912	T_{CFP}	2400 sec
Q	0.453181	$SUCCESS$	1

Table 3. Parameter set for the one-station model.

address the multi-station case. We address both state-based, path-based and nested properties, assuming the set of typical parameter values of the model given in Table 3[2].

Qualitative properties. The following qualitative properties have been verified based on a CTMC derived from a SAN model composed of a single station. As a first functional property we check whether a successful communication can occur after the station has a token on place $FAIL$.

$$@fail \Rightarrow \mathcal{P}_{>0}(\Diamond @succ)$$

This property holds in states M1, M2 and M3 of the CTMC, but not in state M4. So, we can conclude that when a station has a token on place $FAIL$, it can indeed no longer successfully receive further broadcast messages.

The following property states that after $n \leq OD$ failures (i.e. the $COUNTER$ equals n), the station may eventually successfully receive a next broadcast. Since ETMCC checks the property for all states, it is enough to verify

$$(c = n) \Rightarrow \mathcal{P}_{>0}(\Diamond @succ)$$

This property is satisfied by all states. This includes state M2 where the counter is equal to one, i.e. the situation in which one failure has occurred after receiving a successful broadcast.

It can also be shown that after n failures it is still possible to eventually be successful, but this is not guaranteed.

$$(c = n) \Rightarrow \neg \mathcal{P}_{\geq 1}(\Diamond @succ)$$

This property is satisfied by every state.

Quantitative properties. For the verification of the properties in this section the accuracy is set to 10^{-6}.

We have seen that it is not always the case that a failure is followed by a success, so, we could wonder how likely it is that a success takes place in that case. Instead of requiring a

[2]In the work by Coccoli et al. [10, 9, 4] the long duration for T_{CFP} has been chosen because their research has taken place in the context of collaborative mobile robots for which only the CFP phase of the wireless protocol is used [11]. Unfortunately, in their publications they sometimes write 2400 *ms* instead of seconds.

state	probability
M1	0.9133486
M2	0.9311429
M3	0.9526165
M4	1.0

Table 4. Probability to reach FAIL from each state in a single station model

probability greater than or equal to one, i.e. always, we can verify the same formula for a lower probability.

$$(c = 1) \wedge \mathcal{P}_{>0.2}(\Diamond @succ)$$

For the values in Table 3, this property is satisfied by state M2. Note also that this is a state formula with an enclosed path-formula. Such formulas can be easily and conveniently expressed in CSL. They could also be checked using UltraSAN using so called path-automata, but this is much less direct and concise.

$\mathbf{P_{D>OD}}$: The probability that the station after $OD + 1$ retransmissions does not receive the decision message (i.e. reaches the fail state) during the contention free period is less than 0.3 can be formulated as:

$$\mathcal{P}_{<0.3}(\Diamond^{\leq T_{CFP}} @fail)$$

where T_{CFP} is the duration of the CFP equal to 2400 seconds. Table 4 gives the probability for each state to fulfil $\Diamond^{\leq T_{CFP}} @fail$. The results in Table 4 show that the above formula is not satisfied by any state.

Moreover, we can show that the probability to have only successful communications during the period of T_{CFP} is larger than or equal to 0.3. This is captured by the formula

$$\mathcal{P}_{\geqslant 0.3}(\Box^{\leqslant T_{CFP}}(c = 0))$$

which is equivalent to the formula

$$\mathcal{P}_{<0.7}(\text{true } U^{\leqslant T_{CFP}} \neg (c = 0))$$

For $T_{CFP} = 2400$ seconds, this is not satisfied by any station.

5.4. Properties for multi-station models

Of course, what we are really interested in are the results for models with more than one station and for different values of OD. UltraSAN provides two different ways for composing subnets using the repeat operator (REP) or the join operator (JOIN). The REP operator is used to replicate the same subnet a specified number of times. It also allows for the selection of shared places. The advantage of this operator is that it allows for the generation of a reduced CTMC that exploits the symmetry in the specification (RBM). The reduced CTMC is lumping-equivalent to the non-reduced model [12]. In the reduced model the identity of the stations is, however, not maintained. The generated marking file only reports how many stations have which marking. A fragment of an example marking file is given in Fig. 7 (for a case with $N = 4$ and $OD = 6$). It says that for three of

```
MARKING #4
station 3 { COUNTER: 0, FAIL: 0,
            POLL: 1, SUCCESS: 1 }
station 1 { COUNTER: 2, FAIL: 0,
            POLL: 1, SUCCESS: 0 }
```

Figure 7. Marking generated for SAN model with 4 stations composed by REP

the four stations the counter is equal to 0, they did not reach the FAIL place, they are ready to be polled and the previous communication with the AP was successful. One station has had two consecutive failures in communication with the AP, but no final failure, it is ready to be polled and the previous communication with the AP was not successful. For example, for the marking in the reduced model above we encode the results as: s3c0 (i.e., three stations have a counter equal to zero), s1c2 (i.e. one station has a counter equal to two), and so on[3].

The JOIN operator joins subnets while maintaining their identity. In order to join four stations we need to make four copies of the station subnet, each with its own name, and join them together. Also in this case shared places can be selected. No reduction of the state space takes place and the above example, assuming that we have four stations called a, b, c and d, is now represented as in Fig. 8. This leads to labels such as $a1c0$ meaning that station a (of which there is only one) is in a state where its $COUNTER$ has value 0. In this paper we introduce more readable alternative names for these labels in order to improve the readability of the properties. These names will be introduced when they are used.

In both cases it is not difficult to automatically generate a label file for ETMCC by encoding markings into proper atomic propositions.

In the following we address qualitative and quantitative properties of the multi-station model.

Qualitative properties. For the verification of the qual-

[3]The label file format of ETMCC requires that no numbers occur in the names of atomic propositions, therefore, for the actual verification we use labels such as 'sthreeczero' for $s3c0$ and so on.

```
MARKING #4
a 1 { COUNTER: 0, FAIL: 0,
      POLL: 1, SUCCESS: 1 }
b 1 { COUNTER: 0, FAIL: 0,
      POLL: 1, SUCCESS: 1 }
c 1 { COUNTER: 0, FAIL: 0,
      POLL: 1, SUCCESS: 1 }
d 1 { COUNTER: 2, FAIL: 0,
      POLL: 1, SUCCESS: 0 }
```

Figure 8. Marking generated for SAN model with 4 stations composed by JOIN

itative properties we used the CTMC derived from a four station SAN model composed using the JOIN operator. We call the stations a, b, c and d. In the following formulas the variables i and j range over the set of stations. We extend the atomic propositions and write $i@fail$ and $i@succ$ to indicate that station i has a token on place $FAIL$ or on place $SUCCESS$ respectively. Furthermore, we use $c_i = k$ (with k a natural number) to denote that the $COUNTER$ of station i has value k. One of the properties that one would like to hold in the multi-station model is that when one of the stations reaches a failure, it can no longer be involved in successful communications. This property can be formulated as:

$$i@fail \Rightarrow \neg \mathcal{P}_{>0}(\Diamond\ i@succ)$$

We verified the property for $i = a$ and ETMCC reports that it holds for 162 states (out of 189). Verifying the negation of this formula gives the 27 states that do not satisfy the property. More detailed inspection of the markings of those states shows that in all of them both $a@fail$ and $a@succ$ hold. This is explained by the fact that station a is marked as $FAIL$ as a consequence of the failure of another station (remember that the place $FAIL$ is shared among the stations) at the moment that station a just had a successful communication. Moreover, inspecting the file with the model transitions it becomes clear that all the 27 states are final states. So, no further communication can take place from those states.

A similar property shows that when station a reaches a failure and the counter of the omissions is equal to 1 it will remain equal to one forever.

$$i@fail \wedge (c_i = 1) \Rightarrow \mathcal{P}_{\geq 1}(\Box\ c_i = 1)$$

For $i = a$, ETMCC reports that this formula holds for all states.

Another consequence of sharing place $FAIL$ in the model is that when one station fails, all the others fail as well (in the model). This can be formalised for $i, j \in$

$\{a, b, c, d\}$ as:

$$i@fail \Rightarrow \forall i \neq j.j@fail$$

We checked this for $i = a$ with ETMCC, which reports that this formula holds in every state of the full CTMC.

The model of individual stations is, to a limited extent, keeping track of the history of the success or failure of receiving broadcast messages from the AP. If the previous broadcast by the AP to the station was successfully received, it uses one probability distribution for the next success or failure. If it was a failure it uses another distribution. This requires that *in the global model every station* needs to deal with *every* broadcast message sent by the AP. Therefore, the model should not allow traces (paths) in which e.g. only one station deals with the broadcasts, and an other does not perform any transition. Properties like this cannot be formulated for the reduced model, because in that model the identities of the stations are no longer maintained, but they can be formulated for the full state (JOIN) model. For instance, we consider it undesirable if from any state in which the counter of station i is zero and that of station j ($i \neq j$) is one there exists a path in which station j remains in the state with its counter on one (i.e. does not perform any transition) and station i proceeds to a state in which its counter has become 3 (i.e. proving that it made at least 3 transitions). This can be formalised in CSL as:

$$\mathcal{P}_{>0}(\Diamond(c_i = 0 \wedge c_j = 1 \wedge \mathcal{P}_{>0}(c_j = 1 \ \mathcal{U} \ c_i = 3)))$$

where $i, j \in \{a, b, c, d\}$ and $i \neq j$. Model checking for the case that $i = a$ and $j = b$ shows that the formula is satisfied by 81 out of 189 states, including the initial state.

This nested path-based property clearly shows that the current model is not properly capturing the broadcast-aspect of the protocol. In Section 6 we shall therefore propose a new model that does address this aspect properly.

Quantitative properties. Although we have seen from the qualitative properties that this model has some problems, it is nevertheless worth to have a look at the results for quantitative analysis. In particular for the property $P_{D>OD}$, which gives us an occasion to compare the results in the literature obtained with UltraSAN with those obtained with ETMCC. For the verification of quantitative properties we used a CTMC derived from a SAN model with four stations composed by means of the REP operator. The atomic propositions can therefore only address *the number* of stations that are in a state in which variables and places have certain values. Consequently, in the following we slightly change the names of atomic propositions and write $\#@fail = k$ with k a natural number, to indicate that k stations have a token on place $FAIL$.

$\mathbf{P_{D>OD}}$: The property that a station does not receive the decision message after $OD + 1$ retransmissions can be

FDT	T_{CFP}	T_P	T_M	N
3.0E-03	2400 sec.	7.646 ms	2.380 ms	4

PE	P	Q
1.6E-04	0.999871	0.19314
5.0E-04	0.999718	0.43541
1.0E-03	0.999571	0.57104

Table 5. Parameter values for results of Fig. 9

generalised to the multi-station case. First, note that it never happens that a state is reached where two or more stations are in $FAIL$ starting from a state in which no stations were in $FAIL$[4]. This is easy to formulate as:

$$\mathcal{P}_{\geq 1}(\neg(\#@fail = 2 \ \vee \ \#@fail = 3) \ \mathcal{U} \ \#@fail = 1)$$

The formula holds in all states of the model. Of course, in general, it *is* possible that two or more stations are in the fail-state at the same time, which is easily verified by:

$$\#@fail = 2 \ \vee \ \#@fail = 3$$

The above observations allow us to formulate the property about the reception of the decision message as the probability that *one* station reaches the fail-state within the duration of the contention free period:

$$\mathcal{P}_{<\pi}(\Diamond^{\leq T_{CFP}} \ \#@fail = 1)$$

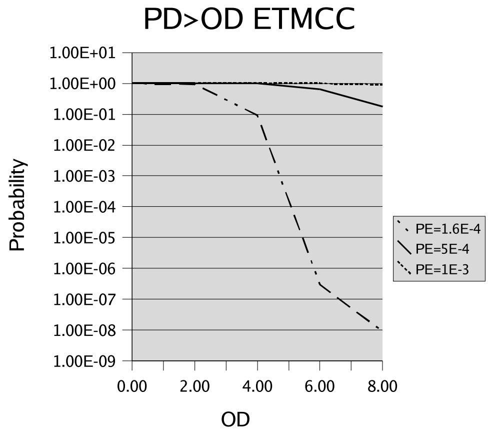
Fig. 9 shows the probabilities that in a system with four stations at least one station does not receive the decision message from the AP due to fading phenomena and under the assumptions and parameters given in Table 5, for omission degrees of 2, 4, 6 and 8 resp., for various packet loss probabilities (PE), with normalized Doppler frequency (FDT) equal to 3.0E-03 and a duration of T_{CFP} equal to 2400 seconds.

The values obtained with UltraSAN correspond quite well to those obtained with ETMCC, except for PE equal to 1.6E-4. For that series ETMCC shows very different outcomes for an omission degree of 6 and higher. This is very likely due to the fact that the model is extremely stiff (i.e. the ration between the largest and the smallest rate in the CTMC is very high.) In fact in the same model we find durations in the order of several milliseconds as well as a duration of 40 minutes for T_{CFP}. The version of ETMCC that we used (v. 1.4.2) has a build-in on-the-fly steady state analysis that is also performed during transient analysis for effi-

[4]Note that the atomic proposition $\#@fail = 4$ is never generated, such a property never occurs in the generated markings in the case of four stations due to the applied reduction algorithm used to generate the reduced base model.

ciency reasons. However, for very stiff models this on-the-fly analysis may lead to the premature detection of a non-existent steady state, and therefore of the incomplete accumulation of probability density. This is also well-illustrated in Fig. 9, were the value obtained for $OD = 6$ in the case of ETMCC is orders of magnitude smaller than expected.

The problem can be easily remediated by switching off (removing) the on-the-fly steady state analysis during transient analysis. In that case renewed analysis shows that the results for ETMCC and UltraSAN correspond as expected for all considered values of OD.



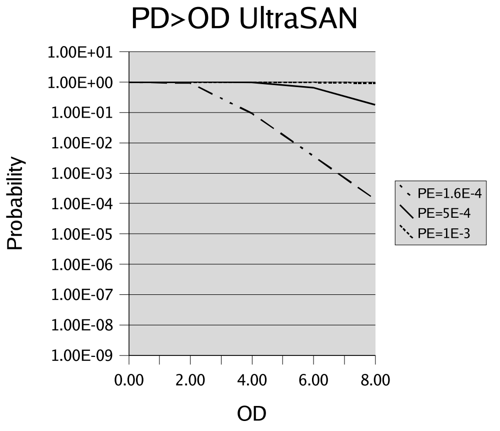


Figure 9. Results obtained with ETMCC and UltraSAN for $P_{D>OD}$ property.

$\mathbf{P_{UM}}$: The property that the AP does not receive acknowledgments for a user message by all the stations within res retransmissions and within T_{CFP} can be formulated in the same way as for property $P_{D>OD}$. In fact, with the modification of the values of the variables P and Q and the rate of the exponential distribution in the way described in Section 4, the same formula in this case reflects the probability that a station does not receive the retransmitted user message for res times in a row:

$$\mathcal{P}_{<\pi}(\Diamond^{\leq T_{CFP}} \#@fail = 1)$$

We postpone the analysis of P_{UM} to the next section, where we develop a more faithfull model of the wireless protocol behaviour.

6. Including Synchrony in the Model

In this section, we develop a new model for the one analysed in the previous section. The key point in the new model is that we want to make sure that every broadcast by the AP to the stations is processed by every station in the model within the same slot. Moreover, we aim again at a model in which all stations are modelled in the same way and can be composed by REP for taking advantage of the reduction strategies of UltraSAN. Finally, it would be preferable not to introduce further subnets in order to avoid the generation of large state-spaces. Therefore, we develop a model that takes care of the synchronisation of stations in a fully distributed way.

6.1. New model

Typically, in the new model, each station processes one broadcast message and then waits until all stations in the group have done so. When the last station has processed the broadcast, it notifies the other stations about this by means of a shared variable. After this the stations are ready for processing the next broadcast from the AP.

We point out that there is no prescribed order in which the stations deal with the broadcast in every slot. This allows for the abstraction of the identity of the stations, which is an advantage for state-space reduction techniques. Secondly, the rate assigned to the timed activity is the same as in the previous model for $P_{D>OD}$. This is allowed because all rates are exponential, so they enjoy the memoryless property. The stations perform their timed activity one after the other in the model. But, by following a similar reasoning as in ([13], p.63), the delay of the second station represents the distribution of the 'remaining delay' after the first station processed its broadcast, which is again exponentially distributed with the same rate as that of the first station. The same holds for any further stations in the group.

In order to model the synchronisation, we need to introduce two more shared places to the model. The place $WAITING$ is a counter that records the number of stations in the slot that have processed the broadcast. The place $TURN$ is a simple boolean that communicates the change of slot to all stations. Both places, and the place $POLL$ are initialised to zero. Each station can perform the timed activity as long as MARK($TURN$)== MARK($POLL$) holds.

At the start of the timed activity the station flips the value of place $POLL$ by means of taking $TURN + 1$ modulo M where M equals 2. After the completion of the timed activity the station behaves as in the previous model, but it also increases the counter $WAITING$ and checks whether it is the last station that performed the timed activity. If so, it flips the value of $TURN$ and resets the value of $WAITING$ to zero. Since now both $POLL$ and $TURN$ have flipped their value, each station is again able to perform the next timed activity, i.e. deal with the next broadcast. Fig. 10 shows the new SAN model and Table 6 gives the definitions of its SAN elements.

In the sequel we call the model of Section 4 model A and the new model described in the current section model B. As before, models for $P_{D>OD}$ and P_{UM} can be obtained by selecting the proper values for the model parameters.

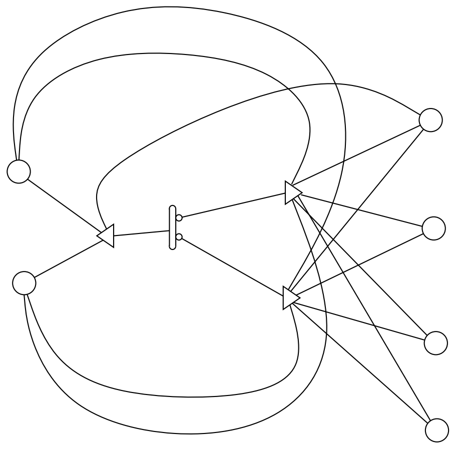


Figure 10. Station in model B

6.2. Properties of the new model

In order to formulate properties for model B, which has additional variables, we introduce the following names for atomic properties much in the same way as we have done for model A. Let i and j range over the set of stations $\{a, b, c\}$ and k a natural number, then we mean by $wait_i = k$ that the value of variable $WAITING$ of station i is equal to k. Further we use $turn_i = k$ ($poll_i = k$) to denote that there is ($k = 1$) or is not ($k = 0$) a token on place $TURN$ ($POLL$).

Qualitative properties. For the verification of the qualitative properties we used a CTMC derived from a SAN model composed of three[5] stations by means of the $JOIN$ operator. For model B we can now verify that the qualitative property of Sect. 5.4, namely

$$\mathcal{P}_{>0}(\Diamond(c_i = 0 \wedge c_j = 1 \wedge \mathcal{P}_{>0}(c_j = 1 \ \mathcal{U} \ c_i = 3)))$$

Timed activity	nprb
Exp. distr. rate	1/(2*GLOBAL_D(TM) + GLOBAL_D(TP))
Case 1	if (MARK(SUCCESS)==1) return (1-GLOBAL_D(P)); else return GLOBAL_D(Q);
Case 2	if (MARK(SUCCESS)==1 return (GLOBAL_D(P)); else return (1-GLOBAL_D(Q));
Output gate	FAIL_BC
Output function	MARK(SUCCESS)=0; MARK(COUNTER)++; if (MARK(COUNTER) > GLOBAL_S(OD)) {MARK(FAIL)=1; } else { MARK(WAITING)++; if (MARK(WAITING) == GLOBAL_S(N)) {MARK(TURN) = (MARK(TURN)+1)%GLOBAL_S(M); MARK(WAITING) = 0;} else ; }
Output gate	SUCC_BC
Output function	MARK(SUCCESS)=1; MARK(COUNTER)=0; MARK(WAITING)++; if (MARK(WAITING) == GLOBAL_S(N)) { MARK(TURN) = (MARK(TURN)+1)%GLOBAL_S(M); MARK(WAITING) = 0; } else ;
Input gate	chk
Input predicate	(MARK(FAIL)==0) && (MARK(POLL)==MARK(TURN))
Input function	MARK(POLL) = (MARK(POLL)+1) % GLOBAL_S(M)
Initial marking	
FAIL	0
COUNTER	0
SUCCESS	1
POLL	0
WAITING	0
TURN	0
M	2

Table 6. Details of elements of model B

[5] We used three stations here because the prototype ETMCC currently allows a maximum of 63 different labels to denote the atomic properties which is less than what is needed in the case of four stations. This limitation will be removed in future versions of ETMCC.

is, as expected, not satisfied by any of the 594 states.

We can also show that whenever for all stations MARK($TURN$)== MARK($POLL$), all stations are ready to start a new slot, i.e. the value of $WAITING$ is equal to 0.

$$\left.\begin{array}{l}(\forall i.turn_i = 1 \wedge poll_i = 1) \vee \\ (\forall i.turn_i = 0 \wedge poll_i = 0)\end{array}\right\} \Rightarrow (\forall i.wait_i = 0)$$

The property is satisfied by all states.

Moreover, just to be sure that the left-hand side of this formula is not trivially satisfied we verify

$$\begin{array}{l}(\forall i.turn_i = 1 \wedge poll_i = 1) \vee \\ (\forall i.turn_i = 0 \wedge poll_i = 0)\end{array}$$

which holds in 54 out of a total of 594 states.

Quantitative properties. At this point we are of course interested in the difference between the results for model A and B. We address the results for $P_{D>OD}$ and P_{UM}. In the following we present only the results obtained with UltraSAN; those obtained with ETMCC coincide to a high degree using the formulas formalizing $P_{D>OD}$ and P_{UM} as introduced in Section 5.4.

$\mathbf{P_{D>OD}}$: Fig. 11 shows the difference between the two models for the case that $PE = 5.0e^{-4}$ and for several values of OD ranging from 0 to 8. It is clear that model B estimates the probability of an error lower than is the case with model A. Of course, in both models, the higher the value of OD the lower the probability of errors.

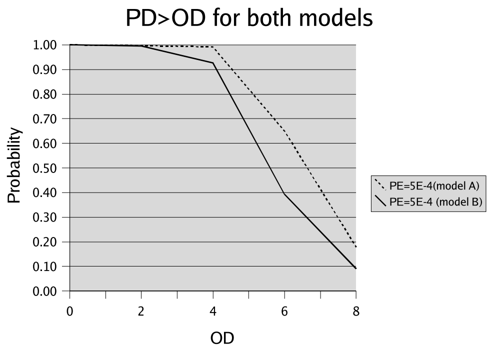


Figure 11. Comparing $P_{D>OD}$ for the interleaving and synchronised model

$\mathbf{P_{UM}}$: Fig. 12 shows several results for the property P_{UM} using the values for the parameters P and Q and the rate of the exponential distribution of the timed activity as described in Sect. 4. The results have been obtained for $PE = 1.6E - 4$ and the resilience res varying between $[0, .., 3]$. The upper two curves are the results for T_{CFP} equal to 2400 seconds. The upper of the two giving the results for model A and the lower of the two those for model B. The third curve from above shows the *experimental* data obtained for a similar setting, as has been reported in [10]. Actually, the value measured for $res = 3$ was equal to 0, so the value could not be established with sufficient precision in Fig. 12. Below the third curve, two curves show the results for model A and B resp., but for T_{CFP} equal to 2400 *milli*seconds, similar to those reported in [4]. Finally, the curve at the bottom shows the results of an earlier model developed by Coccoli et al. [10] that did not consider correlation between communication failures due to fading effects.

Model A, 4 stations, RBM				
OD	2	4	6	8
#States	25	105	249	660
#Transitions	70	385	1260	3135

Model A, 4 stations, full state space				
OD	2	4	6	8
#States	189	1125	3773	9477
#Transitions	713	5369	20313	54953

Model B, 4 stations, RBM				
OD	2	4	6	8
#States	334	1730	5460	13260
#Transitions	784	4840	16800	43320

Model B, 4 stations, full state space				
OD	2	4	6	8
#States	4158	26750	93982	243486
#Transitions	12096	88000		

Table 7. Size of the state-space for various models

6.3. Some statistics

In order to give an idea of the size of the state-space of the models that we have analysed in this paper we give an overview of the number of states and transitions of the underlying CTMC's in Table 7.

Note that the number of states and transitions are independent of the values for P and Q in the models, but these numbers do have influence on the stiffness of the models.

6.4. Discussion

It is clear that neither model A nor model B are matching exactly the experimental data, although model B gives a better approximation than model A. For small values of res (i.e. $res = 0$ or $res = 1$) both models considerably over-estimate the probability of error, while the prediction becomes better for higher values of res ($res = 2$), even if there is not enough experimental data available to give a well-informed judgement. Maybe that the correlation between transmission errors during the experiments was lower than that assumed for the model, or, more likely, an explanation could be that user messages are retransmitted only once per round and are therefore much less susceptible to the bursty nature of a fading channel. In other words, the loss of *user messages* is much less correlated than for example, the loss of consecutive decision messages. Unfortunately, the number of lost *decision messages* has not been established in an experimental way.

Nevertheless, model B is preferred over an earlier model developed in [10] that did not account for fading effects, because the latter considerably under-estimates the error probabilities [10].

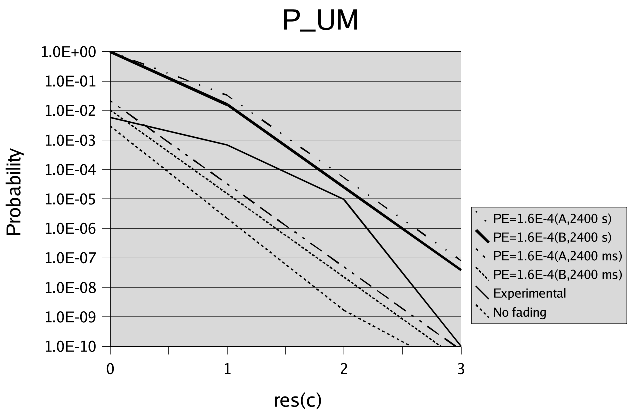


Figure 12. Comparing the results for P_{UM}

7. Conclusions

In this paper we have revisited an earlier dependability analysis of a variant of the centralised medium access protocol of the IEEE 802.11 standard for wireless local area networks [9]. We have analysed some of the models used in that work both from a behavioural (qualitative) and from a dependability (quantitative) point of view by means of the prototype stochastic model checker ETMCC. Both qualitative and quantitative properties have been formlised using CSL.

The qualitative analysis of the concurrent behaviour of the models showed a discrepancy between the expected behaviour of the model and its actual behaviour. The use of model checking allowed for the clear and unambiguous specification and verification of the desired behavioural properties. Some of these concerned properties over state *sequences*, that can in general only be analysed in an indirect way by means of path-automata by current state-of-the-art dependability analysis tools such as UltraSAN or Möbius [6]. Extending these tools with (stochastic) model checking capabilities would allow model developers to assess also the often intricate concurrent behaviour of dependability models.

Based on the results of the qualitative analysis we have developed a new model, that has been shown to reflect more faithfully the assumed synchronisation aspects of the protocol that is induced by the concept of broadcasts within single slots and rounds of the protocol.

Two of the main dependability measures, introduced informally in [9], have been formalised as formulas of the Continuous Stochastic Logic and assessed by means of the stochastic model checker ETMCC. The results have been compared with those obtained by means of standard transient analysis of the same models using UltraSAN. The results coincided when on-the-fly steady state analysis is removed from the transient analysis in ETMCC.

We believe that this paper provides further evidence of the potential advantages of the integration of (stochastic) model checking capabilities and advanced tools for model-based dependability and performance analysis and its application to realistic case-studies.

In this paper we have discussed only part of the interesting properties and models for the analysis of the real-time wireless protocol. Our future research aims at a more complete formal analysis of the protocol, using proper abstraction techniques and forms of compositionality in order to address further qualitative and quantitative properties in a coherent and systematic way.

8. Acknowledgments

The authors would like to thank Felicita Di Giandomenico and Andrea Coccoli for sharing with us their models, experimental data and tools and for the clarifying discussions on earlier results, to Salem Derisavi for fruitful discussions and to Holger Hermanns and Joachim Meyer-Kayser for their assistance with ETMCC.

Mieke Massink and Diego Latella have been partially supported by Projects EU-IST IST-2001-32747 (AGILE), MIUR "Special Fund for the Development of Strategic Research" under CNR project "Instruments, Environments and Innovative Applications for the Information Society", sub-project "Software Architecture for High Quality Services for Global Computing on Cooperative Wide Area Networks" and the "Fondo Ricerca Individuale" of CNR-ISTI. Joost-Pieter Katoen has been supported by the HAAST project which is funded by the Dutch Technology Foundation (STW).

References

[1] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous time Markov chains. *ACM Transactions on Computational Logic*, **1**(1): 162–170, 2000.

[2] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *Concurrency Theory*, volume 1664 of LNCS, pp. 146–162, Springer-Verlag, 1999.

[3] C. Baier, B. Haverkort, H. Hermanns and J-P. Katoen. Automated performance and dependability evaluation using model checking. In *Computer Performance Evaluation*, Springer, Rome, Italy, pp. 261-289, 2002.

[4] A. Bondavalli, A. Coccoli and F. Di Giandomenico. QoS analysis of group communication protocols in wireless environment. In *P. Ezhilchelvan and A. Romanovsky (eds.), Concurrency in Dependable Computing*, Kluwer Academic Publishers, The Netherlands, pp. 169-188, 2002.

[5] P. Buchholz, J.-P. Katoen, P. Kemper and C. Tepper. Model-checking large structured Markov chains. *Journal of Logic and Algebraic Programming*, **56**:69–96, 2003.

[6] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders and P. Webster. The Möbius framework and its implementation, *IEEE Trans. Soft. Eng.*, **28**(10):956–969, 2002.

[7] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. Model Checking. MIT Press, Cambridge, MA, 1999.

[8] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. In *ACM Transactions on Programming Languages and Systems*, **8**: 244–263, 1986.

[9] A. Coccoli, A. Bondavalli and F. Di Giandomenico. Analysis and estimation of the quality of service of group communication protocols In *Proc. 4th IEEE Int'l Symp. on Object-oriented Real-time Distributed Computing (ISORC'01)*, pp. 209–216, Magdeburg, Germany, May 2001.

[10] A. Coccoli, S. Schemmer, F. Di Giandomenico, M. Mock and A. Bondavalli. Analysis of Group Communication Protocols to Assess Quality of Service Properties. In: *5th IEEE International Symposium on High Assurance Systems Engineering (HASE 2000)*, Albuquerque, New Mexico, 2000.

[11] A. Coccoli. Personal communication, September 2003.

[12] S. Derisavi. Personal communication, October 2003.

[13] H. Hermanns. *Interactive Markov Chains, and the Quest for Quantified Quality.* Volume 2428 of LNCS, Springer-Verlag Berlin Heidelberg, 2002.

[14] H. Hermanns, J-P. Katoen, J. Meyer-Kayser and M. Siegle. A tool for model-checking Markov chains. *International Journal on Software Tools for Technology Transfer*, Vol. 4, no. 2, pp. 153–172, 2003.

[15] IEEE 802.11. *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE, 1997.

[16] M. Kwiatkowska, G. Norman and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In J.-P. Katoen, P. Stevens (Eds.), *Proc. of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of LNCS, Spinger-Verlag, pp. 52-66, 2002.

[17] M. Kwiatkowska, G. Norman and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala (Eds.), *2nd Joint Int. Workshop on Process Algebra and Performance Modelling and Probabilistic Methods in Verification*, volume 2399 of LNCS, pp. 169-187, Springer-Verlag, 2002.

[18] J. F. Meyer and W. H. Sanders. Specification and construction of performability models, In *Proceedings of the Second International Workshop on Performability Modeling of Computer and Communication Systems*, Mont. Saint-Michel, France, June 28-30, 1993.

[19] A. Mitschele-Thiel. *Systems Engineering with SDL: Developing Performance-Critical Communication Systems*. John Wiley & Sons, 2001.

[20] M. Mock, E. Nett and S. Schemmer. Efficient reliable real-time group communication for wireless local area networks In J. Hlavicka, E. Maehle and A. Pataricza (Eds.), *Proceedings of EDCC-3, The Third European Dependable Computing Conference*, volume 1667 of LNCS, 2000.

[21] W. H. Sanders and J. F. Meyer. Reduced Base Model Construction Methods for Stochastic Activity Networks, In IEEE *Journal on Selected Areas in Communications*, vol. 9, no. 1, pp. 25-36, 1991.

[22] W. H. Sanders, W. D. Obal, M. A. Qureshi and F. K. Widjanarko. The UltraSAN Modeling Environment. *Performance Evaluation*, Vol. 24, pp. 89-115, 1995.

[23] S. Schemmer. Zuverlässige Echtzeit-Gruppenkommunikation auf Einem Lokalen Funknetz. *GMD Research Series*, no. 4, (ISBN 3-88457-376-4), 2000.

[24] H.L.S. Younes and R.G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma and K.G. Larsen (Eds.), *Proc. of the 14th International Conference on Computer Aided Verification*, volume 2404 of LNCS, Springer-Verlag, pp. 223-235, 2002.

[25] M. Zorzi, R. R. Rao and L. B. Milstein. On the accuracy of a first-order Markov model for data block transmission on fading channels, In *Proc. IEEE Int. Conf. on Universal Personal Communications (ICUP'95)*, pp. 211-215, 1995.