Performance Analysis of ARQ Protocols using a Theorem Prover

Osman Hasan
Dept. of Electrical and Computer Engineering
Concordia University
Montreal, Quebec, Canada
o_hasan@ece.concordia.ca

Sofiène Tahar
Dept. of Electrical and Computer Engineering
Concordia University
Montreal, Quebec, Canada
tahar@ece.concordia.ca

Abstract

Automatic-repeat-request (ARQ) protocols are widely used in modern data communications to guarantee reliable transmission over imperfect physical links. The behavior of an ARQ protocol largely depends on a number of network parameters and traditionally simulation is used for their performance analysis. However, simulation provides less accurate results and usually requires enormous amount of CPU time in order to attain reasonable estimates. To overcome these limitations, we propose to conduct the performance analysis of ARQ protocols in the environment of a higher-order-logic theorem prover (HOL). We present an approach to formally model the delay characteristics of ARQ protocols as a function of Geometric random variable in higher-order-logic. In particular, we develop higher-order-logic models that describe the delay behavior of three basic types of ARQ protocols, i.e., Stop-and-Wait, Go-Back-N and Selective-Repeat. The paper also includes the verification of the average message delay relations for these three protocols in HOL.

1 Introduction

Automatic-repeat-request (ARQ) [20], which utilizes the principles of error detection and retransmission, is one of the most widely used techniques for reliable communication between computers. Indeed, it is one of the most important part of the Transmission Control Protocol (TCP) and the High-level Data Link Control (HDLC) standard. In an ARQ system, both sending and receiving stations have error detection capabilities. Thus, the receiver discards the block of data (or frame) received in error, and requests a retransmission of the same frame via a feedback channel. The sender, on the other hand, keeps on retransmitting a frame until and unless it receives an error-free acknowledgement (ACK) of the reception of this frame from the receiver.

The ARQ principle has been implemented using a number of different approaches, usually termed as ARQ protocols. Due to the large variation in implementation complexity and performance, depending on the link quality, performance analysis is very important for the right choice of ARQ protocol for a particular implementation. The performance analysis of ARQ protocols is usually based on probabilistic techniques. The main idea is to model the error in frame transmission by an appropriate random variable and then predict the average delay to transmit a single frame successfully. Obviously, the ARQ protocol that minimizes this delay is termed as the most efficient one.

Since the development of the probability theory in the last century, engineers have been using paper-and-pencil proof techniques to perform statistical analysis for telecommunication systems. These traditional techniques may lead to erroneous results with ARQ protocols as they usually involve a subtle interaction of a number of distributed components and have a high degree of parallelism [4]. The advent of fast and inexpensive computational power in the last two decades opened up avenues for using computers in this domain. Nowadays, computer based simulation approaches [2] are quite frequently used to validate the performance evaluation results of ARQ protocols obtained via paper-and-pencil proof methods. Most simulation software provide a programming environment for defining functions that approximate random variables for probability distributions. The frame transmission error can be modeled by one of these functions and the delay of an ARQ protocol for transmitting a frame successfully can then be predicted using computer simulation techniques, such as the Monte Carlo Method [22], where the main idea is to approximately answer a query on a probability distribution by analyzing a large number of samples. Due to the inherent nature of simulation based analysis, the use of pseudorandom numbers [18] and the approximation introduced by precision effects in computer arithmetic, the simulation results can be quite unreliable at times. It is a common occurrence that different simulation packages come up with different solutions to the same problem. In [23], McCullough proposed

a collection of intermediate-level tests for assessing the numerical reliability of simulation based statistical packages and uncovered flaws in most of the mainframe softwares [24]. These unreliable results pose a more serious problem when it comes to the analysis of ARQ protocols since they are often used in safety critical applications, such as space travel, medicine and transportation, where a mismatch between the predicted and the actual system performance may even result in a loss of human life. Another major limitation of simulation based performance analysis is the enormous amount of CPU time requirement for attaining meaningful estimates. This approach generally requires hundreds of thousands of simulations to calculate the statistical quantities and becomes impractical when each simulation step involves extensive computations.

As an alternative to simulation techniques, we propose to use higher-order logic interactive theorem proving [9] for performance analysis of ARQ protocols. Higher-order logic is a system of deduction with a precise semantics and due to its high expressibility it can be used to model any system that can be expressed in a closed mathematical form. Higher-order-logic has also been successfully used to develop some of the classical mathematical theories. Interactive theorem proving is the field of computer science and mathematical logic concerned with computer based formal proof tools that require some sort of human assistance. The core of theorem provers usually consists of a handful of axioms and primitive inference rules. Soundness is assured as every new theorem must be created from these basic axioms and primitive inference rules or any other already proved theorems or inference rules. Powerful mathematical techniques such as induction and abstraction are the strengths of theorem proving and make it a very flexible verification technique. In this paper, we illustrate the process of expressing the delay characteristics of ARQ protocols in higher-order-logic and verifying their average delay relations as mathematical theorems within the sound core of a theorem prover. The performance analysis carried out in this way will overcome the shortcomings of simulation as it will be free from approximation and precision issues, and the accuracy of the results will be independent of the CPU time.

The foremost requirement for conducting average delay analysis of ARQ protocols in a higher-order-logic theorem prover is to have access to the mathematical theories of natural and real numbers, sets and probability. The HOL system [11], which is a higher-order-logic theorem prover, comes with a very rich built-in library of mathematical theories including the ones mentioned above. Therefore, we propose to use HOL and its built-in libraries to formally express the delay characteristic of ARQ protocols as a function of the Geometric random variable. These formal expressions can in turn be used, along with the higher-order-logic formalization of the expectation (or average) function given in [13], to verify the average delay relations for the corresponding protocols in HOL. In order to illustrate the practical effectiveness of our approach, we present the verification of average delay relations for three of the most commonly used ARQ protocols, i.e., Stop-and-Wait, Go-Back-N and Selective-Repeat in HOL. To the best of our knowledge, this is the first time such analysis is done using a higher-order-logic theorem prover.

The rest of the paper is organized as follows: After reviewing the related work in Section 2, we proceed by presenting an informal description of the three ARQ protocols under consideration in Section 3. Next, Section 4 provides an overview of modeling random variables and verifying their probabilistic and expectation properties in HOL. In Section 5, we present a formal probabilistic model of the number of frame retransmissions required to transmit a frame successfully using ARQ in terms of the probability that the transmission of a single bit is in error. We utilize formally specified Bernoulli(p) and Geometric(p) random variables for this purpose. In Section 6, we present the HOL verification of an expectation property that allows us to evaluate the expectation value of a random variable X multiplied and added by two constants in terms of the expectation of the random variable X. This property, sometimes referred to as the linearity of expectation, plays a vital role in the verification of average message delay relations of ARQ protocols. Then, in Section 7, we utilize the formalization and verification done so far to formally model the delay characteristics of Stop-and-Wait, Go-Back-N and Selective-Repeat protocols and verify the corresponding average values in HOL. Finally, some concluding remarks are given in Section 8. The HOL symbols, used in this paper, and their corresponding mathematical interpretations are given in the Appendix at the end.

2 Related Work

Work on the performance analysis of ARQ protocols has always existed since the early days of their introduction, however, using theoretical paper-and-pencil proofs and simulation techniques. Several ARQ protocols have been analyzed and compared mostly through their average delay for a successful transmission [21]. More specifically, Fayolle *et al.* [8] analyzed the Stop-and-Wait, Towsley and Wolf [30] analyzed the standard Go-Back-N and Easton [7] analyzed a Selective-Repeat protocol. One of the most widely used simulation based software for network protocol performance analysis is OPNET [26], which has been used to validate the paper-and-pencil performance analysis proofs for different ARQ schemes, e.g., [15, 31]. In this paper, we show how the performance analysis of ARQ protocols can be mechanically performed using the HOL theorem prover, providing a superior approach to validation by simulation.

Theorem provers have also been used for the verification of ARQ protocols. For example, Cardell-Oliver [3] and Chkliaev *et al.* [4] presented the verification of the sliding window protocol, which is a general idea and covers the principles of most of the commonly used ARQ protocols, using HOL and PVS [27], respectively. But, to the best of our knowledge, all the research available in this domain is related to functional verification and no work related to the performance evaluation of an ARQ protocol using a theorem prover can be found in the open literature.

The foremost criteria for conducting formalized performance analysis using a theorem prover is to be able to formalize random variables and reason about their expectation properties in higher-order logic. Hurd's PhD thesis [16] can be considered a pioneering work in this regard as it presents a formalization of probability theory in HOL. Using his mathematical theories, Hurd was able to formalize and verify a few discrete random variables. Building upon Hurd's formalization framework, the work in [13] presented the formalization of some expectation theory, which allows the verification of the expected (or average) values associated with discrete random variables that attain values in *positive integers* only. We mainly build upon the above mentioned libraries in HOL and extend them by verifying an expectation property that allows us to express the expectation of a random variable of the form $a+bX$, where a and b are any *positive integers*, in terms of the expectation of the random variable X. This property plays an important role in the verification of the average delay relations for ARQ protocols as will be illustrated in Section 7 of this paper.

Another alternative for formal performance analysis of network protocols is to use probabilistic model checking techniques, e.g., [1, 28]. It is an automated verification technique that involves the construction of a precise mathematical model of the probabilistic system which is then subjected to exhaustive analysis to verify if it satisfies a set of formal properties. Expectation is one of the most useful tools in performance analysis and therefore its evaluation within a model checker is being explored in the probabilistic model checking community. Some probabilistic model checkers, such as PRISM [19] and VESTA [29], offer the capability of verifying expected values in a semi-formal manner. For example, in the PRISM model checker, the basic idea is to augment probabilistic models with cost or rewards: real values associated with certain states or transitions of the model. This way, the expected value properties, related to these rewards, can be analyzed by PRISM. Duflot *et al.* [6] used this aspect of PRISM to conduct the performance analysis of a CSMA/CD protocol. It is important to note that the meaning ascribed to these properties is, of course, dependent on the definitions of the rewards themselves and thus the solutions obtained using this kind of approaches are approximate as has been clearly stated in [6]. On the other hand, there is no such risk involved in verifying the expectation properties using the proposed theorem proving based approach due to its inherent soundness. Other major limitations of probabilistic model checking include the inability to handle systems that do not exhibit the memory-less property and the well-known state space explosion problem [5]. In contrast, higher-order-logic theorem proving is capable of handling all kinds of systems as long as they can be expressed in a closed mathematical form due to the high expressibility of higher-order logic.

3 ARQ Protocols

There are three basic types of ARQ protocols: Stop-and-Wait, Go-Back-N and Selective-Repeat. In this section, we provide a brief overview of these three protocols in terms of their delay characteristics. Details about other issues such as sequence numbering, etc. can be found in [20]. The operation is described under the assumptions that both sending and receiving stations have error detection capabilities and all information and ACK frames have the same length.

3.1 Stop-and-Wait ARQ

Stop-and-Wait is the most simplest ARQ protocol that allows the transmission of only one frame at a time.

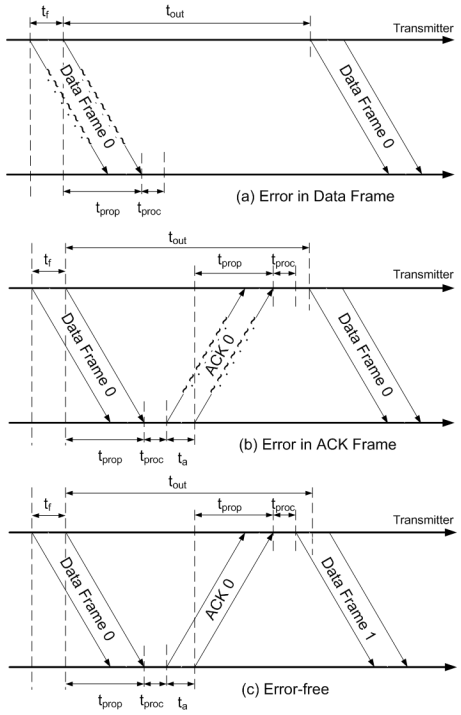

Figure 1. Stop-and-Wait ARQ Operation

The transmitter sends a single information frame of n_f bits to the receiver and spends t_f time units in doing so. It then stops and waits to receive an ACK from the receiver. If no ACK is received within a given time out, t_{out}, period, the frame is resent by the transmitter and once again it stops and waits for the ACK (Figure 1.a). If an ACK is received within the given t_{out} period then the transmitter checks the received frame for errors during the next t_{proc} time units. If errors are detected then the ACK is ignored and the frame is resent by the transmitter after t_{out} expires and once again the transmitter stops and waits for the ACK (Figure 1.b). If no errors are detected in the ACK frame then the transmitter transmits the next frame in its queue (Figure 1.c).

The receiver is always waiting to receive information frames. When a new frame arrives, the receiver checks it for errors during the next t_{proc} time units. If errors are detected then the information frame is ignored and the receiver continues to be in the wait state (Figure 1.a), otherwise it initiates the transmission of an ACK frame of n_a bits, which takes t_a time units (Figure 1.b,c).

Under the above mentioned conditions, the ACK message cannot be received before $t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc}$ units of time pass after sending out an information frame, where t_{prop} represents the one-way propagation time between transmitter and receiver. It is, therefore, necessary to set the time out period t_{out} to be greater than or equal to $2(t_{prop} + t_{proc}) + t_a$ for reliable communication.

The performance of the Stop-and-Wait protocol is expected to be quite low as both the transmitter and receiver are left idle for some time, even for the cases when there are no errors. But on the other hand, the simplicity of the approach translates into a cheap hardware implementation.

3.2 Go-Back-N ARQ

The Go-Back-N protocol tends to increase the efficiency of Stop-and-Wait by introducing parallelism in the frame transmission. The operation is illustrated in Figure 2.

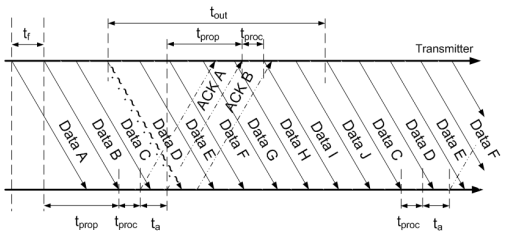

Figure 2. Go-Back-N ARQ Operation

The transmitter continuously transmits information frames without waiting for ACKs. The window size, which represents the limit on the number of frames at the transmitter that can be outstanding without receiving an ACK, is usually chosen larger than $\frac{t_{out}}{t_f}$ to ensure that the channel is always kept full. For transmitting a single frame of n_f bits, it takes t_f time units. If no ACK is received within the t_{out} period for a given frame, the protocol starts retransmitting all the frames starting from this frame, as indicated in Figure 2 for the case of data frame C. On the other hand, if an ACK is received within the given t_{out} period, then the transmitter checks the received frame for errors during the next t_{proc} time units. If errors are detected or there is a problem with the sequence numbers [20], then the ACK is ignored. When, the t_{out} of the corresponding frame expires, the protocol begins retransmitting all the frames starting from this frame just like it does when no ACK is received.

The receiver is always waiting to receive information frames. When a new frame arrives, the receiver checks it for errors during the next t_{proc} time units. If errors are detected or there is a problem with sequencing numbers, then the information frame is ignored and the receiver continues to be in the wait state, otherwise it initiates the transmission of an ACK frame, which takes t_a time units.

It is important to note that due to the pipeline effect in the Go-Back-N protocol, new information frames are being transmitted and received during the processing and propagation periods and the time spent on transmitting the ACK. Thus t_{prop}, t_{proc} and t_a do not contribute to the net delay. Though, the condition $t_{out} \geq 2(t_{prop}+t_{proc})+t_a$ still needs to hold for Go-Back-N like the Stop-and-Wait protocol.

3.3 Selective-Repeat ARQ

Selective-Repeat is among the most powerful ARQ based protocols. Both the transmitter and the receiver have a packet buffer of at least the current window size and thus can retain out-of-sequence packets, rather than discarding them as in the Go-Back-N protocol. Though, this added benefit comes at the cost of computational power that is required at the receiver to reorder the incoming packets in sequence and the buffering, which can become quite expensive for channels having long propagation delays.

The actions of the transmitter in Selective-Repeat protocol are similar to the ones in Go-Back-N except the fact that the transmitter retransmits only the frame that is required to be transmitted again rather than all the frames starting from this frame. A frame is required to be transmitted again if its corresponding ACK has not been received within the t_{out} period or was found to be in error. The operation of the receiver in Selective-Repeat is also similar to the case of Go-Back-N except for the fact that it keeps track of all the data frames that are received without error irrespective of their sequence numbers. It is important to note here that in this particular version of Selective-Repeat protocol the transmitter needs to receive an ACK for every information frame it

sends, i.e., it cannot deduce the successful transmission of a frame by receiving an ACK message for a later frame.

4 Random Variables and their Expectation

Random variables can be formalized in higher-order logic as deterministic functions with access to an infinite Boolean sequence $\mathbb{B}^\infty$; a source of infinite random bits [16]. These deterministic functions make random choices based on the result of popping the top most bit in the infinite Boolean sequence and may pop as many random bits as they need for their computation. When the functions terminate, they return the result along with the remaining portion of the infinite Boolean sequence to be used by other programs. Thus, a random variable which takes a parameter of type α and ranges over values of type β can be represented in HOL by the function.

$$\mathcal{F} : \alpha \rightarrow B^\infty \rightarrow \beta \times B^\infty$$

As an example, consider the $Bernoulli(\frac{1}{2})$ random variable that returns 1 or 0 with equal probability $\frac{1}{2}$. It can be formalized in HOL as follows

```
⊢ bit = λs.if shd s then 1 else 0,stl s
```

where *s* is the infinite Boolean sequence and `shd` and `stl` are the sequence equivalents of the list operation *'head'* and *'tail'*. [16] also presents some formalization of the mathematical measure theory in HOL, which can be used to define a probability function $\mathbb{P}$ from sets of infinite Boolean sequences to $real$ numbers between 0 and 1. The domain of $\mathbb{P}$ is the set $\mathcal{E}$ of events of the probability. Both $\mathbb{P}$ and $\mathcal{E}$ are defined using the Carathéodory's Extension theorem, which ensures that $\mathcal{E}$ is a σ-algebra: closed under complements and countable unions. The formalized $\mathbb{P}$ and $\mathcal{E}$ can be used to formally verify probabilistic properties, e.g.,

```
⊢ ℙ {s | fst(bit s) = 1} = 1/2
```

where the HOL function `fst` selects the first component of a pair and $\{x|C(x)\}$ represents a set of all x that satisfy the condition C.

The above approach has been successfully used to formalize and verify both discrete [16, 13] and continuous random variables [12] in HOL. In this paper, we utilize the models for Bernoulli and Geometric random variables formalized as higher-order-logic functions `ber_rv` and `geo_rv` and verified using the following probability mass function (PMF) relations in [16] and [13], respectively.

Theorem 1: *PMF of Bernoulli(p) Random Variable*

```
⊢ ∀ p.  0 ≤ p ∧ p ≤ 1 ⇒
  ℙ{s | fst(ber_rv p s)} = p
```

Theorem 2: *PMF of Geometric(p) Random Variable*

```
⊢ ∀ n p.  0 < p ∧ p ≤ 1 ⇒
  ℙ{s | fst(geo_rv p s)=(n+1)} = p(1-p)^n
```

The Geometric random variable returns the number of Bernoulli trials needed to get one success and thus cannot return 0. This is why we have *(n+1)* in Theorem 2, where *n* is a positive integer $\{0, 1, 2, 3 \cdots\}$. Similarly, the probability *p* in Theorem 2 represents the probability of success and thus needs to be greater than 0 for this theorem to be true as has been specified in the precondition.

Expectation theory plays a vital role in the domain of probabilistic performance analysis as it is a lot easier to judge performance issues based on the average value of a random variable, which is a single number, rather than its distribution function. [13] presents a higher-order-logic definition of the expectation function for discrete random variables that attain values in positive integers only

Definition 1: *Expectation of Discrete Random Variables*

```
⊢ ∀ R. expec R =
  suminf (λn.nℙ{s | fst(R s)=n})
```

where, `suminf` represents the HOL formalization of the infinite summation of a *real* sequence [10] as outlined in Appendix A. The function `expec` accepts the random variable R with data type $B^\infty \rightarrow (positive\ integer \times B^\infty)$, and returns a $real$ number. The above definition can be used to verify the average values of most of the commonly used discrete random variables, e.g., [13] presents the verification of average value of the Geometric random variable.

Theorem 3: *Average of Geometric(p) Random Variable*

```
⊢ ∀ n p.  0 < p ∧ p ≤ 1 ⇒
  (expec (λs.geo_rv p s) = 1/p)
```

In order to target the verification of expected values of probabilistic systems involving multiple random variables, [13] presents the formal proof of the generic linearity of expectation property [17] for addition in HOL. By this property, the expectation of a sum of random variables equals the sum of their individual expectations

$$Ex[\sum_{i=1}^{n} R_i] = \sum_{i=1}^{n} Ex[R_i] \tag{1}$$

where Ex denotes expectation.

5 Number of Frame Retransmissions in ARQ

In this section, we formalize the number of retransmissions required to transmit a frame successfully using the ARQ protocols, described in Section 3, in terms of the bit-error probability of the channel. This representation will be

used to formally express the delay characteristic of ARQ protocols later.

The number of retransmissions of a frame in these ARQ protocols is a random quantity that is dependant on the event when both the data frame and the corresponding ACK frame propagate through the channel without incurring any errors. We keep on retransmitting the given frame until the above event occurs. Under the assumption that each event of a frame being corrupted while propagating through the channel is independent of any other such event, we can formally model the number of retransmissions using the Geometric random variable [20]. We already have a higher-order-logic function `geo_rv` for the Geometric random variable and thus the next step towards the formal representation of the number of retransmissions is to find the corresponding success probability.

Assuming that the bit-error probability for the given channel is known and is equal to p, we can model the error behavior of a single bit during transmission as a Bernoulli(p) random variable. Now, a frame incurs errors if one or more of its bits incur errors. Thus, the error behavior of the whole frame during transmission can be modeled as the following higher-order-logic function.

Definition 2: *Frame Error Event*

```
⊢ ∀ n p s.  f_err 0 p s = (𝔽,s) ∧
  f_err (n+1) p s = ((fst(ber_rv p s) ∨
    (fst(f_err n p (snd(ber_rv p s))))),
    (snd(f_err n p (snd (ber_rv p s)))))
```

where the symbol $\mathbb{F}$ represents logical $False$. The function `f_err` recursively models a random variable in higher-order logic that accepts a positive integer n, a real number p and the infinite Boolean sequence s and returns a pair with the first component as a Boolean and the second component as the remaining portion of the infinite Boolean sequence. The functions `fst` and `snd` return the first and the second components of a pair. It is important to note that each new call of the function `ber_rv` in Definition 2 uses the remaining portion of the infinite Boolean sequence from the previous call. This fact ensures that the events involving the function `ber_rv` in Definition 2 are independent.

The first component of the random variable `f_err` is *true*, if the value of any of the n Bernoulli random variables represented by the function `ber_rv` is *true*. Therefore, the function `f_err` models the error behavior of an n bit frame transmission over a channel with bit-error rate equal to p as each Bernoulli random variable models the behavior of a single bit transmission.

Next, we utilize the random variable `f_err` to formalize the probability of success for the Geometric random variable that in turn can be used to model the number of retransmissions for the ARQ protocols of Section 3. The probability of success in this case is equal to the probability of the event when both the information frame and its corresponding ACK frame do not incur errors during transmission. This probability can be formally expressed as follows:

Definition 3: *Probability of Successful Transmission*

```
⊢ ∀ nf na p.  suc_p_arq nf na p =
ℙ{s | fst(f_err nf p s) ∨
fst(f_err na p (snd(f_err nf p s))) = 𝔽}
```

The function `suc_p_arq` accepts two *positive integers* nf and na that correspond to the frame lengths of information and ACK frames, respectively, and a real number p that represents the bit-error rate of the communication channel. It returns the probability of successful transmission of the information frame. Again, it is important to note that the second call of the random variable `f_err` utilizes the remaining portion of the infinite Boolean sequence after the first call, which ensures that the events corresponding to these two calls are independent. Next, we verify the following equivalent form for the `suc_p_arq` function in HOL.

Theorem 4: *Probability of Successful Transmission*

```
⊢ ∀ nf na p.  0 ≤ p ∧ p ≤ 1 ⇒
  suc_p_arq nf na p = (1-p)^(nf+na)
```

We proceed with the proof of the above theorem by expressing its left hand side (LHS) in terms of the function `f_err`

```
⊢ ∀ nf na p.  suc_p_arq nf na p =
  ℙ{s | fst(f_err (nf + na) p s) = 𝔽}
```

using the function definitions of `suc_p_arq` and `f_err` along with some simplification based on Boolean logic and set theory axioms. This expression allows us to generalize the statement of Theorem 4 as follows

```
⊢ ∀ n p.  0 ≤ p ∧ p ≤ 1 ⇒
  ℙ{s | fst(f_err n p s) = 𝔽} = (1-p)^n
```

The above proof goal can now be verified in HOL using induction on its *positive integer* variable n. The base case, i.e., `n = 0`, is $true$ since both the right hand side (RHS) and the LHS of the above subgoal are equal to 1. Whereas, in the step case, we get the following subgoal using the function definition of `f_err`

```
0 ≤ p ∧ p ≤ 1 ∧
ℙ{s | fst(f_err n p s) = 𝔽} = (1-p)^n ⇒
  ℙ{s | (fst (ber_rv p s)=𝔽) ∧
    (fst(f_err n p (ber_rv p s))=𝔽)}
  = (1-p)^(n+1)
```

The LHS of the above subgoal can now be simplified using the independence property of the events involved, the classical probability law $\mathbb{P}(AB) = \mathbb{P}(A)\mathbb{P}(B)$ and some set theory principles formalized in HOL as follows

```
ℙ{s | (fst (ber_rv p s) = 𝔽) ∧
   (fst(f_err n p (ber_rv p s)) = 𝔽)} =
ℙ{s | fst(ber_rv p s)=𝔽}
    ℙ{s | fst(f_err n p s)=𝔽}
```

which in turn allows us to verify the proof goal of the above step case, since we know from Theorem 1 that `ℙ{s | (fst(ber_rv p s) = 𝔽)} = (1-p)`, and thus concludes the proof of Theorem 4.

The number of retransmissions for the ARQ protocols, outlined in Section 3, is one less than the value of the Geometric random variable with the success probability given in Theorem 4. This is the case because the formalized Geometric random variable returns the least number of trials required to obtain the first success. Thus, we need to subtract 1 form this number, in order to ignore the trial for the final successful transmission. The number of retransmissions can now be used to express the delay characteristic of an ARQ protocol as a random variable in HOL.

$$(\lambda \mathtt{s}.\mathtt{T_u}(\mathtt{fst}(\mathtt{geo\_rv}(\mathtt{suc\_p\_arq}\ \mathtt{n_f}\ \mathtt{n_a}\ \mathtt{p})\mathtt{s}) - \mathtt{1}) + \mathtt{T_s},\ \mathtt{snd}(\mathtt{geo\_rv}(\mathtt{suc\_p\_arq}\ \mathtt{n_f}\ \mathtt{n_a}\ \mathtt{p})\mathtt{s})) \tag{2}$$

where T_s and T_u are *positive integers* that represent the time required for the final successful transmission of one frame and the total time wasted during a single unsuccessful transmission of a frame, respectively. It is important to note here that for the analysis, presented in this paper, we have chosen a discrete time domain, i.e., time is represented by *positive integers* rather than *real* numbers. This choice reduces the complexity of the verification task by a considerable extent without compromising on the reliability of the analysis. These *positive integers* may be thought to be representing the ticks of a clock counting physical time in any appropriate units, e.g., nanoseconds. Whereas, the granularity of the clock's tick is believed to be chosen in such a way that it is sufficiently fine to detect properties of interest.

6 Verification of Linearity of Expectation Property in HOL

In this section, we present the HOL verification of the linearity of expectation property, which allows us to evaluate the expectation value of a random variable R multiplied and added by two *positive integers* a and b, respectively, in terms of the expectation of the random variable R.

$$Ex[aR + b] = aEx[R] + b \tag{3}$$

The above equation plays a significant role in the verification of the average message delay characteristic of ARQ protocols, since the random variable that models the ARQ message delay characteristic, given in Equation 2, is of the format $aR + b$.

We proceed with the HOL proof of the linearity property, given in Equation 3, by first verifying

$$Ex[aR] = aEx[R] \tag{4}$$

which allows us to evaluate the expectation value of a random variable R multiplied by a *positive integer* a in terms of the expectation of the random variable R. The property, given in Equation 4, can be expressed in HOL as follows

```
⊢ ∀a R. expec(λs.(a fst(R s),snd(R s)))
  = a(expec R)
```

for a random variable `R` with a *well-defined* expected value, i.e., the summation corresponding to the expectation for this random variable is convergent. The HOL proof proceeds by first performing case analysis on the variable `a`, which is basically a *positive integer*. For the case when `a` is `0`, the RHS of the proof goal becomes 0. Whereas, using the definition of expectation, the LHS reduces to the expression

$$\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\mathtt{0} = \mathtt{n}\})$$

which is also equal to 0 as `∀n.n ℙ{s | 0 = n} = 0`. On the other hand, when `a` is not equal to `0`, i.e., $(0 < a)$, the proof goal may be simplified as

$$\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\mathtt{a}\,\mathtt{fst}(\mathtt{R}\,\mathtt{s}) = \mathtt{n}\}) =$$

$$\mathtt{a}\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\mathtt{a}\,\mathtt{fst}(\mathtt{R}\,\mathtt{s}) = \mathtt{a}\,\mathtt{n}\})$$

by using the definition of expectation and some arithmetic reasoning. Next, we prove in HOL that

$$\forall \mathtt{k}.(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\ \mathtt{a}\,\mathtt{fst}(\mathtt{R}\,\mathtt{s}) = \mathtt{n}\}) =$$

$$\mathtt{a}(\sum_{\mathtt{n}=0}^{\mathtt{B(k)}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\ \mathtt{a}\,\mathtt{fst}(\mathtt{R}\,\mathtt{s}) = \mathtt{a}\,\mathtt{n}\})$$

where `B(k) = if (k MOD a = 0) then (k DIV a) else ((k DIV a) + 1)` and `MOD` and `DIV` represent the *modulo* and *division* functions for positive integers in HOL. This allows us to rewrite our proof goal as follows

$$\lim_{\mathtt{k}\to\infty} \mathtt{a}\,(\sum_{\mathtt{n}=0}^{\mathtt{B(k)}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\mathtt{a}\,\mathtt{fst}(\mathtt{R}\,\mathtt{s}) = \mathtt{a}\,\mathtt{n}\}) =$$

$$\mathtt{a}\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathtt{n}\,\mathbb{P}\{\mathtt{s}|\mathtt{a}\,\mathtt{fst}(\mathtt{R}\,\mathtt{s}) = \mathtt{a}\,\mathtt{n}\})$$

which can be proved using the properties of limit of a real sequence in HOL [10], since both of the real sequences in the above equation converge to the same value as the value of k becomes very very large. This concludes the proof of the expectation property given in Equation 4.

The expectation properties given in Equations 1 and 4 can now be combined to verify the linearity of expectation property given in Equation 3 and the result can be expressed in HOL as

Theorem 5: *Linearity of Expectation*

```
⊢ ∀ a b R. expec
   (λs.(a fst(R s) + snd(R s)))
     = a(expec R) + b
```

for a random variable `R` with a *well-defined* expected value.

7 Average Delay Characteristic of ARQ Protocols

In this section, we utilize the formalization and the verification presented above to formally verify the average delay relations of the three ARQ protocols described in Section 3. The main idea is to plug in the values of T_s and T_u for each one of these protocols one by one in the delay random variable, given in Equations 2, and then utilize the linearity of expectation property, formally verified in Theorem 5, to verify the corresponding average values in HOL.

7.1 Stop-and-Wait ARQ

In the Stop-and-Wait protocol, the time, T_s, required for a successful transmission is equal to $t_f + t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc}$ time units as can be seen from Section 3.1. On the other hand, T_u, that is the time required for each unsuccessful transmission attempt for Stop-and-Wait protocol is equal to $t_f + t_{out}$ time units. We can now formalize the message delay random variable of the Stop-and-Wait protocol using the above mentioned approach as follows.

Definition 4: *Stop-and-Wait Protocol Delay*

```
⊢ ∀ tout tprop tproc p tf ta nf na.
sw_del tout tprop tproc p tf ta nf na=
(λs.(tf + tout)
   (fst(geo_rv(suc_p_arq nf na p) s) - 1)
     + tf + ta + 2(tproc + tprop),
   snd(geo_rv (suc_p_arq nf na p) s))
```

The function `sw_del` accepts the delay parameters for the Stop-and-Wait protocol and specifies its delay characteristic as a higher-order-logic random variable. The average message delay of the Stop-and-Wait protocol can now be verified in HOL using the random variable `sw_del` and Theorem 5, if the expectation relation for the random variable $(\lambda s.(\texttt{fst}(\texttt{geo\_rv}\ (\texttt{suc\_p\_arq}\ \texttt{nf}\ \texttt{na}\ \texttt{p})\ \texttt{s}) - 1, \texttt{snd}(\texttt{geo\_rv}\ (\texttt{suc\_p\_arq}\ \texttt{nf}\ \texttt{na}\ \texttt{p})\ \texttt{s})))$ is known. Using the definitions of expectation, given in Definition 1, and the Geometric random variable `geo_rv`, given in [13], and the probability theory principles, formalized in [16], we verified the following theorem in this regard in HOL

Theorem 6: *Average of (Geometric(p)-1)*

```
⊢ ∀ p.  0 < p ∧ p ≤ 1 ⇒
  expec (λs.(fst(geo_rv p s)-1,
    snd(geo_rv p s))) = (1−p)/p
```

Now, we are in the position of verifying the average message delay relation for the Stop-and-Wait protocol

Theorem 7: *Average Delay of Stop-and-Wait Protocol*

```
⊢ ∀ tout tprop tproc p tf ta nf na.
0 ≤ p ∧ p < 1 ⇒
expec
(sw_del tout tprop tproc p tf ta nf na)
  = (tf + tout) (1−(1−p)^(nf+na))/(1−p)^(n.f+n.a) +
    tf + ta + 2(tproc + tprop)
```

The HOL proof is based on Theorems 4, 5 and 6 along with some arithmetic reasoning.

7.2 Go-Back-N ARQ

In the Go-Back-N protocol, T_s is equal to t_f as can be seen from Section 3.2. It is important to note that the time $t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc}$ is not considered a part of T_s because during this time other frames are being transmitted in parallel, i.e, it is being utilized to do other useful work. On the other hand, T_u in this case is still equal to $t_f + t_{out}$, since the transmitter has to wait t_{out} time units, after transmitting a data frame, before it knows that the data was not properly delivered. Like in the case of the Stop-and-Wait protocol, the message delay characteristic of the Go-Back-N protocol can now be formalized as follows.

Definition 5: *Go-Back-N Protocol Delay*

```
⊢ ∀ tout p tf nf na.
gbn_del tout p tf nf na =
(λs.(tf + tout)
   (fst(geo_rv(suc_p_arq nf na p) s) - 1)
     + tf,
   snd(geo_rv (suc_p_arq nf na p) s))
```

Again, using Theorems 4, 5, and 6 along with some arithmetic reasoning, we were able to prove the following average delay relation for the Go-Back-N protocol in HOL.

Theorem 8: *Average Delay of Go-Back-N Protocol*

```
⊢ ∀ tout p tf nf na.  0 ≤ p ∧ p < 1 ⇒
  expec (gbn_del tout p tf nf na)
    = (tf + tout) (1−(1−p)^(nf+na))/(1−p)^(n.f+n.a) + tf
```

7.3 Selective-Repeat ARQ

In the Selective-Repeat protocol, the time required for the final successful transmission of a frame, T_s, and the time required for each unsuccessful transmission attempt, T_u, are both equal to t_f (Section 3.3). This is the case because during all other delays such as the propagation or processing of frames, other frames are being transmitted in parallel and thus that time is being utilized in a useful manner. The average delay characteristic of the Selective-Repeat protocol can now be formalized as follows.

Definition 6: *Selective-Repeat Protocol Delay*

```
⊢ ∀ p tf nf na.
sr_del p tf nf na =
(λs.tf
  (fst(geo_rv(suc_p_arq nf na p) s) - 1)
    + tf,
  snd (geo_rv(suc_p_arq nf na p) s))
```

Again, using Theorems 4, 5 and 6 along with some arithmetic reasoning, we were able to prove the following average delay relation for the Selective-Repeat protocol.

Theorem 9: *Average Delay of Selective-Repeat Protocol*

$$\vdash \forall\, p\ tf\ nf\ na.\quad 0 \leq p \wedge p < 1 \Rightarrow \mathtt{expec\ (sr\_del\ p\ tf\ nf\ na)} = \frac{\mathtt{tf}}{(1-\mathtt{p})^{(\mathtt{n.f}+\mathtt{n.a})}}$$

The above exercise illustrates the fact that interactive theorem proving is capable of conducting performance analysis of ARQ protocols with at least the same degree of accuracy as the analytical proof techniques usually carried out using paper-and-pencil proof methods; a novelty that cannot be achieved by any other computer based techniques, such as simulation or model checking. As outlined before in this paper as well, simulation based techniques are based on many approximations and thus can never achieve accurate results. Similarly, due to the inherent limitations of the state-based formal methods, discussed in Section 2, they cannot evaluate the expected values as precisely as we have attained using the proposed approach for the message delay characteristic of ARQ protocols.

On the other hand, the proposed approach is also superior than the paper-and-pencil proof methods in a way as the chances of making human errors and proving wrongful statements are almost nil since all proof steps are applied by the computer. For this argument to hold, correctness of the checking machinery must of course be ensured. In the case of the HOL system, this property is ensured by construction following the so-called LCF paradigm. Milner [25] developed the underlying design philosophy, which employs a small trusted logical kernel to flexibly implement any logic that is built on top by reducing any proof to the simple deduction steps the LCF kernel allows.

8 Conclusions

In this paper, we utilized the mathematical probability theory formalized in a higher-order-logic theorem prover to verify the average delay of three basic types of ARQ protocols. During this process, we presented a formalization of the delay characteristic of ARQ protocols as a higher-order-logic random variable and the verification of linearity of expectation property for discrete random variables. To the best of our knowledge, this is the first study on using these kind of techniques for such an application. Due to the formal nature of the models and the inherent soundness of theorem proving systems, the analysis is guaranteed to provide exact answers. This feature makes the proposed approach very useful for the performance optimization of safety critical and highly sensitive telecommunication protocols.

Our approach for the performance analysis of ARQ protocols is quite general and can be extended and easily adapted to conduct precise performance analysis of other network protocols or computer algorithms as well. The random or unpredictable elements, such as noise, found in the analysis of telecommunication protocols can be modeled using an appropriate random variable from the existing library of formalized discrete [16, 13] and continuous random variables [12], and the precise average values associated with the parameters of interest may then be verified within the sound core of a higher-order-logic theorem prover. The verification of other statistical properties such as variance and tail bounds, which provide further insight into the performance issues, may be included in the analysis, based on the formalization presented in [14].

The main limitation of the proposed approach is the associated significant user interaction, i.e., the user needs to guide the proof tools manually since we are dealing with higher-order logic, which is known to be non-decidable. Because of this, the proposed approach should not be viewed as an alternative to methods such as simulation and model-checking for the performance analysis of real-time systems but rather as a complementary technique, which can prove to be very useful when precision of the results is of prime importance.

References

[1] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen. Model Checking Algorithms for Continuous time Markov Chains. *IEEE Trans. on Software Engineering*, 29(4):524–541, 2003.

[2] P. Bratley, B. Fox, and L. Schrage. *A Guide to Simulation*. Springer-Verlag, 1987.

[3] R. Cardell-Oliver. *The Formal Verification of Hard Real-time Systems*. PhD Thesis, University of Cambridge, Cambridge, UK, 1992.

[4] D. Chkliaev, J. Hooman, and E. de Vink. Verification and Improvement of the Sliding Window Protocol. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2619 of *LNCS*, pages 113–127. Springer, 2003.

[5] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 2000.

[6] M. Duflot, L. Fribourg, T. Hérault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic Model Checking of the CSMA/CD Protocol using PRISM and APMC. In *Proc. 4th Workshop on Automated Verification of Critical Systems*, pages 195–214. Elsevier Science, 2004.

[7] M. Easton. Batch Throughput Efficiency of ADCCP/HDLC/SDLC Selective Reject Protocols. *IEEE Transactions on Communications*, 28(2):187–195, 1980.

[8] G. Fayolle, E. Gelenbe, and G. Pujolle. An Analytic Evaluation of the Performance of the "Send and Wait" Protocol. *IEEE Transactions on Communications*, 26(3):313–319, 1978.

[9] M. Gordon. Mechanizing Programming Logics in Higher-0rder Logic. In *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer, 1989.

[10] J. Harrison. *Theorem Proving with the Real Numbers*. Springer, 1998.

[11] J. Harrison, K. Slind, and R. Arthan. HOL. In *The Seventeen Provers of the World*, volume 3600 of *LNCS*, pages 11–19. Springer, 2006.

[12] O. Hasan and S. Tahar. Formalization of the Continuous Probability Distributions. In *Automated Deduction*, volume 4603 of *LNAI*, pages 3–18. Springer, 2007.

[13] O. Hasan and S. Tahar. Verification of Expectation Properties for Discrete Random Variables in HOL. In *Theorem Proving in Higher-Order Logics*, volume 4732 of *LNCS*, pages 119–134. Springer, 2007.

[14] O. Hasan and S. Tahar. Verification of Tail Distribution Bounds in a Theorem Prover. In *Numerical Analysis and Applied Mathematics*, volume 936, pages 259–262. American Institute of Physics, 2007.

[15] F. Hou, P. Ho, and Y. Zhang. Performance Analysis of Differentiated ARQ Scheme for Video Transmission over Wireless Networks. In *Proc. 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 1–7. ACM Press, 2005.

[16] J. Hurd. *Formal Verification of Probabilistic Algorithms*. PhD Thesis, University of Cambridge, Cambridge, UK, 2002.

[17] R. Khazanie. *Basic Probability Theory and Applications*. Goodyear, 1976.

[18] D. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley Professional, 1998.

[19] M. Kwiatkowska, G. Norman, and D. Parker. Quantitative Analysis with the Probabilistic Model Checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2):5–31, 2005. Elsevier.

[20] A. Leon Garcia and I. Widjaja. *Communication Networks: Fundamental Concepts and Key Architectures*. McGraw-Hill, 2004.

[21] S. Lin, D. Costello, and M. Miller. Automatic-Repeat-Request Error-Control Schemes. *IEEE Communications Magazine*, 22(12):5–17, 1984.

[22] D. MacKay. Introduction to Monte Carlo Methods. In *Learning in Graphical Models, NATO Science Series*, pages 175–204. Kluwer Academic Press, 1998.

[23] B. McCullough. Assessing the Reliability of Statistical Software: Part I. *The American Statistician*, 52(4):358–366, 1998.

[24] B. McCullough. Assessing the Reliability of Statistical Software: Part II. *The American Statistician*, 53(2):149–159, 1999.

[25] R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17:348–375, 1977.

[26] OPNET. http://www.opnet.com/, 2008.

[27] PVS. http://pvs.csl.sri.com, 2008.

[28] J. Rutten, M. Kwaiatkowska, G. Normal, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilisitc Systems*, volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[29] K. Sen, M. Viswanathan, and G. Agha. VESTA: A Statistical Model-Checker and Analyzer for Probabilistic Systems. In *Proc. IEEE International Conference on the Quantitative Evaluation of Systems*, pages 251–252, 2005.

[30] D. Towsley and J. Wolf. On the Statistical Analysis of Queue Lengths and Waiting Times for Statistical Multiplexers with ARQ Retransmission Schemes. *IEEE Trans. on Communications*, 27(4):693–702, 1979.

[31] J. Wall and J. Khan. An ARQ enhancement with QoS Support for the 802.11 MAC Protocol. In *Proc. IEEE Conference on Wireless Communications and Networking*, volume 1, pages 430–435, 2004.

A HOL Symbols

HOL Symbol	**Meaning**
$\wedge$	Logical and
$\vee$	Logical or
λx.t	Function that maps x to $t(x)$
{x|P(x)}	Set of all x such that $P(x)$
(a, b)	A pair of two elements
fst	First component of a pair
snd	Second component of a pair
suminf(f)	Infinite summation of a $real$ sequence $\lim_{k\to\infty} \sum_{n=0}^{k} f(n)$