Hindawi Publishing Corporation
International Journal of Vehicular Technology
Volume 2010, Article ID 238518, 23 pages
doi:10.1155/2010/238518

Research Article

Verification of the FlexRay Transport Protocol for AUTOSAR In-Vehicle Communications

Steven Gordon and San Choosang

Sirindhorn International Institute of Technology, Thammasat University, 131 Moo 5, Tiwanont Road, Muang, Pathumthani 12000, Thailand

Correspondence should be addressed to Steven Gordon, steve@siit.tu.ac.th

Received 15 July 2010; Revised 18 October 2010; Accepted 3 November 2010

Academic Editor: Martin Reisslein

The FlexRay Transport Protocol (FrTp) is designed to support reliable and efficient communication between various computers embedded in vehicles. It uses a standardised FlexRay communication bus and introduces a go-back-N style retransmission algorithm. A formal modelling language, Coloured Petri nets (CPN), has been applied to verify the protocol design. Separate CPN models of the FrTp service and protocol are developed and with state space analysis-used to prove for selected configurations that FrTp is deadlock-free and conforms to the service specification when transferring a single-protocol data unit from sender to receiver. In addition, closed-form solutions relating the state space size, retransmission limit, and number of segments are found, giving increased confidence that FrTp is error-free, even for configurations where the state explosion problem arises.

1. Introduction

Vehicles today may contain nearly 100 embedded computers, or Electronic Control Units (ECUs), that together control the engine, airbags, suspension, seats, as well as provide information to other on-board devices and users (e.g., telephone and entertainment systems) [1]. The increasing number and complexity of applications using ECUs has contributed to the development of new in-vehicle communication systems, as well as techniques to simplify the development of applications utilising multiple ECUs. For the former, FlexRay [2] is a potential replacement for Controller Area Network (CAN), a bus for inter-ECU communication in many vehicles today. For the latter, automobile manufacturers are developing the Automotive Open Systems Architecture (AUTOSAR) [3] to allow software components to communicate irrespective of the ECU or bus technology (FlexRay, CAN) that is in use. A specific part of AUTOSAR is the FlexRay Transport Protocol (FrTp) [4], which overcomes a limitation of the FlexRay bus [5] in allowing software components to send long messages with high reliability. As future in-vehicle and ITS applications will depend on FrTp for reliable communication, it is important that the design is proved to be free of significant errors. Any unexpected behaviour in vehicle communication systems may lead to expensive bug fixes, recalls, or even fatalities. Formal methods are well suited for the specification and analysis of such safety-critical systems. Coloured Petri nets (CPNs) [6] are a formal modelling language with a graphical notation to ease model development, backed by a mathematical definition that supports automatic proof of properties using state space exploration. This paper applies CPNs to present the first formal verification of the AUTOSAR FlexRay Transport Protocol. Key contributions are the following.

(1) Formal specification of both FrTp protocol and service using Coloured Petri nets. In this paper the CPN model is used for formal analysis of the protocol. However, in future work the CPN model could also be used in other protocol development tasks. For example, with minor modifications to the CPN model, performance analysis of FrTp could be performed or source code could be generated.

(2) Definition of the key desired properties of FrTp, in particular the set of expected terminal markings and the desired service language.

(3) Proof that for selected configurations, when delivering a single-protocol data unit from sender to receiver FrTp is deadlock-free and satisfies the desired service language. Verification of FrTp for these configurations provides confidence that the protocol specification does not contain functional errors.

(4) Closed-form solutions for the size of the FrTp state space as a function of certain protocol parameters (e.g., retransmission and block size limits). The observed trends in the state space size provide increased confidence that the desired properties hold for configurations not yet analysed, thereby avoiding limitations of state space analysis.

An earlier version of the FrTp protocol CPN and analysis results was presented in [7]. This paper presents an updated, more detailed CPN of the FrTp protocol, and also presents a CPN of the FrTp service (which was not in [7]). This paper also includes additional analysis results, in particular verifying the protocol operation for a larger set of configurations and showing the relationship between state space size and protocol parameters.

This paper is structured as follows. Section 2 provides background material on AUTOSAR, FlexRay and protocol verification, and reviews the state of the art in this area. Section 3 provides details of the FlexRay Transport Protocol features. Section 4 introduces CPNs. Section 5 presents our CPN model of the service and protocol. Analysis results proving the correctness of the protocol, as well as other insights, are given in Section 6, followed by concluding remarks in Section 7.

2. Background and Related Work

2.1. AUTOSAR and FlexRay. Future in-vehicle software applications will be implemented as multiple components coordinating with each other to achieve the desired software functionality. Components may be running on different ECUs, interact with various sensors and actuators, and communicate via one or more buses. AUTOSAR [3] is an architecture that aims to simplify the development of component-based software applications by hiding the complexities of the ECU's and communication bus from components.

As an example, consider Figure 1 which shows an application comprised of four components distributed across three ECUs. As part of the AUTOSAR architecture, the Run Time Environment (RTE) provides an API to the software components for communicating and using the underlying operating system and hardware. The RTE provides the ability to startup/shutdown software components, and hides the ECU details from components. Below the RTE are a set of AUTOSAR services, and corresponding hardware abstractions and drivers, that will be commonly used by multiple software applications, thereby avoiding the need to be implemented in each component. These include services for memory management, communications, and I/O, as well as device-specific drivers for the microcontroller and sensors/actuators. The microcontrollers of each ECU then communicate via one of the possible bus technologies.

The focus of this paper is on the communication mechanisms in AUTOSAR, especially when FlexRay is the chosen bus standard. The detailed AUTOSAR layers in this case are shown in Figure 2. Data to be sent between software components is delivered by the AUTOSAR Communication (COM) layer to the PDU Router. AUTOSAR COM provides a communication API for applications and is based upon OSEK/VDX COM [8]. The PDU Router determines the technique to transport data between software components, for example, selecting from the different bus standards (FlexRay, CAN), and based on the service required (reliable versus unreliable).

If FlexRay is selected by the PDU Router, then the FlexRay Transport Protocol (FrTp) may be used to provide reliability and efficiency features not offered by the FlexRay bus. Below the communication services is the FlexRay interface which provides an abstract interface to the FlexRay drivers and FlexRay controller (which may be implemented internal or external to the microcontroller). FlexRay [2] defines a physical and data link layer protocol for distributed bus-based communication between a set of controllers. The physical layer allows each controller to connect via one or two channels at data rates of 10 Mb/s. The data link layer uses TDMA allowing each controller to transmit 255 bytes of data in a frame per time slot.

The focus of this paper is on the FlexRay Transport Protocol within the AUTOSAR architecture. Many of the blocks in AUTOSAR are based on existing standards/protocols/software and have been used/tested extensively. FrTp is in fact based on an existing CAN/ISO transport protocol [9]. However several significant extensions are introduced which warrants in-depth, formal analysis of its operation.

2.2. Protocol Verification. Formal methods can be used for the specification and analysis of communication protocols to provide insights into the system behaviour, early detection of errors in the design process, remove ambiguities from specifications and to prove correctness of the protocol [10]. Formal methods can be applied during various phases of a protocol design: in this paper we concentrate on protocol verification [11, 12], which involves proving the protocol satisfies a set of requirements or desired properties. Two sets of properties of the protocol are considered.

Firstly, there are dynamic properties of a protocol that are expected from its correct operation. Common desired properties include the absence of deadlocks and livelocks in the protocol. Other properties may be specific to the protocol, such as the number of messages stored in a receive buffer never goes above a predefined limit. With a formal, executable model of a protocol, state space analysis is well-suited to automatically proving a desired set of protocol properties.

The second set of properties of interest are the interactions between the protocol and the user, where the user may be another protocol, a software component or (less likely) a human. The users requirements of the protocol, or *service*

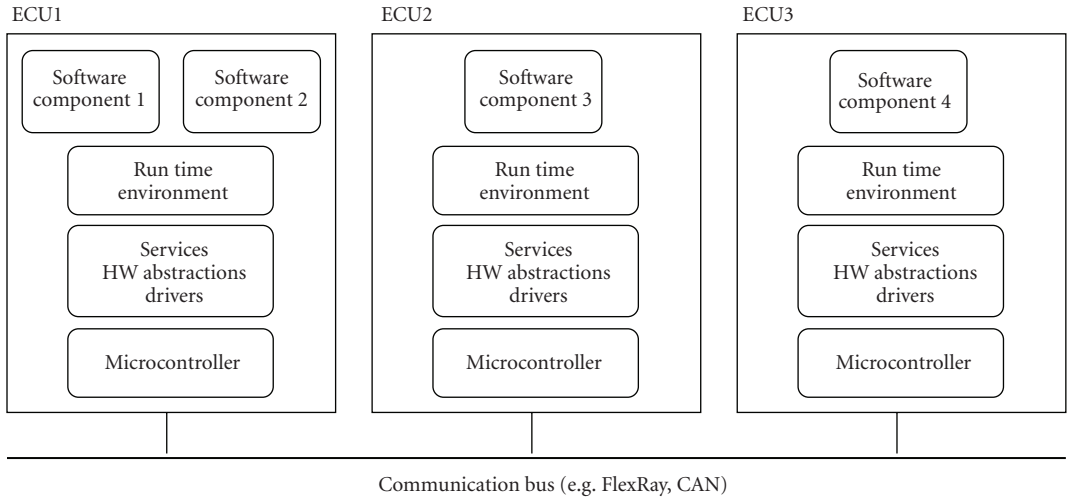

Figure 1: Structure of ECU's in AUTOSAR.

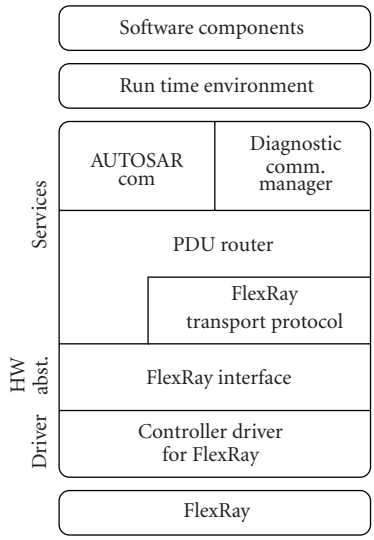

Figure 2: AUTOSAR FlexRay layered architecture.

specification, should define the set of possible interactions between user and protocol. The interactions are referred to as *service primitives* and the set of possible orderings of service primitives is the *service language*. Protocol verification aims to prove that the protocol faithfully implements the desired service specification. Again, with a formal model of a protocol, as well as the desired service language, state space analysis combined with language theory can be applied to verify that the sequence of service primitives allowed by the protocol faithfully refines that of the service language.

In this paper Coloured Petri nets [6] are used to model FrTp, as well as the FrTp service specification. CPNs are selected as the formal modelling language as they: support features for modelling protocols, that is, concurrency, nondeterminism; allow for modelling at different levels of abstraction, that is, service and protocol; have a graphical notation, simplifying the validation of the model; and have computer tool support for modelling and state space analysis (CPN Tools [13]).

2.3. Related Work. The development of an automotive embedded system consists of multiple phases, ranging from requirements specification through to conformance testing. Reference [14] identifies the importance of system level models and verification of functional and nonfunctional properties during the development of automotive embedded systems, including AUTOSAR. An example of validating requirements are met across all phases in AUTOSAR development is shown in [15]. Recently, researchers have addressed specific development phases for AUTOSAR.

As many AUTOSAR applications will be time-critical, verifying that timing constraints are met is important. Reference [16] describes a case study of applying the TIMMO methodology and the related modelling language TADL to analyse the timing of an AUTOSAR steer-by-wire application using the FlexRay bus. The authors demonstrate that TIMMO/TADL is well suited to the AUTOSAR development process and can be used to verify solutions meet timing requirements. Reference [17] analyses the AUTOSAR operating system for the cases when faulty designs, in particular underestimation of Worst Case Execution Time (WCET), leads to timing errors. The authors show the AUTOSAR timing protection mechanisms achieve the goal of avoiding propagation of faults, but in some cases is poor at reducing the number of jobs missing deadlines. An approach for containing timing errors in one AUTOSAR component so that it does not impact on other components is presented in [18]. A synchronization protocol is proposed to allow the different components to access shared resources.

Generating code from system configuration information and models can increase confidence that the resulting code is error-free. Reference [19] shows how an executable RTE can be generated from XML descriptor files for AUTOSAR. Reference [20] presents a methodology for generating the AUTOSAR services (below the RTE), using the FlexRay communication services as an example.

Conformance testing is another phase in ensuring a design/implementation meets requirements. Reference [21] extends TTCN-3 with additional real-time features to allow testing of specific timing scenarios within AUTOSAR. Reference [22] presents an example of testing AUTOSAR software generated from a model-based design.

Although there has been no formal analysis of the communication protocols used in AUTOSAR, there exists a wide body of paper in protocol verification in general (see, e.g., [10] and the references within). FrTp uses a go-back-N error-control mechanism: numerous verification studies of go-back-N [23–25] and the simpler stop-and-wait protocol [26–29] have been conducted. One of the most recent, and closest to our work is [29], which presents extensive verification of a class of stop-and-wait protocols using CPNs. From state space and language analysis, properties are proved for selected configurations of stop-and-wait. Then, through detailed inspection of the CPN, the analysis is extended to prove the properties for arbitrary parameter values. Our work follows a similar approach, but differs in that we consider a different, more complex protocol (go-back-N, instead of stop-and-wait; modelling of features specific to FlexRay/AUTOSAR) and include detailed modelling of the FrTp interaction with the higher layer. Although we only prove properties for specific parameter values (not arbitrary values as in [29]), we do provide analysis that increases our confidence that FrTp is error-free for all configurations. Note also that although FrTp uses go-back-N principles, there are substantial differences to the protocols analysed in [23–25] (e.g., types of frames, interactions with higher layer and parameters) such that independent verification is warranted for FrTp.

Reference [30] presents a formal model of the CAN bus, which can be used as an alternative to FlexRay in AUTOSAR. A CPN model is created and used to measure throughput and latency in a simple CAN network. However [30] is not attempting formal verification of the protocol, nor focussing on AUTOSAR or FlexRay.

3. FlexRay Transport Protocol in AUTOSAR

The FlexRay Transport Protocol (FrTp) provides both a confirmed and unconfirmed communication service for AUTOSAR applications that use a FlexRay bus. It is partly based on ISO 15765-2 [9], a standard for unconfirmed communication for diagnostic applications in a CAN-based vehicle. In addition FrTp provides extra reliability features (e.g., acknowledgements and retries) to offer confirmed communication. Segmentation and flow control are also implemented to improve performance. The error/flow control scheme is based on go-back-N. This paper, and subsequent description, focusses on the confirmed service, as it is substantially different from and more complex than the existing ISO 15765-2 protocol. Only features relevant to the modelling/analysis tasks are described in this section; for a full treatment of FrTp see [4].

3.1. FrTp Service Specification. Application and/or diagnostics data is sent via AUTOSAR COM or DCM to the PDU Router. If required, the PDU Router delivers that data to FrTp, which then transfers the data to the destination ECU, PDU Router and eventually application. The service provided by FrTp to the user (PDU Router) is informally described in Section 5 of [4]. A set of service primitives are defined, as shown in Figure 3. (In [4] each primitive is prefixed with a layer name, e.g., FrTp_Transmit. For brevity, in Figure 3 and subsequent discussion we denote primitives in italics and omit the layer name, e.g., *Transmit*.)

The key operations for data transfer are the following.

(1) *Transmit* primitive is called by PDU Router to initiate data transmission at the sender.

(2) The FlexRay Interface (FrIf) is used to transmit that data to the receiver FrTp.

(3) Upon successfully receiving the data, the receiver passes that data to the PDU Router via *RxIndication* primitive. In addition a confirmation is sent to the sender and passed to the PDU Router as *TxConfirmation* with the *successful* flag set.

(4) Unsuccessful delivery of data results in no indication to the receiver PDU Router, but a *TxConfirmation* with *unsuccessful* flag set is delivered to sender PDU Router.

Other primitives are used for optional features such as cancelling a transmission.

Although [4] defines the set of service primitives, there is no formal definition of the valid sequences of primitives. In Section 5.1 we develop a formal model of the FrTp service, then use it to generate the set of possible service primitive sequences, that is, the service language.

3.2. FrTp Protocol Specification. The FrTp protocol specification is given in Sections 7, 8, and 9 of [4]. Data that arrives from the higher layer, which is referred to as a Protocol Data Unit (PDU), can be sent using different methods with FrTp:

(1) unacknowledged segmented or unsegmented transfer,

(2) acknowledged segmented or unsegmented transfer, but no retries,

(3) acknowledged segmented or unsegmented transfer with retries,

(4) unacknowledged segmented or unsegmented multicast transfer.

The first method is identical to ISO 15765-2, while the other three are new. This paper addresses only the unicast acknowledged transfer (methods 2 and 3), leaving multicast transfer for future work.

Different frame types are used in FrTp. All frames contain the addresses of the sender and receiver. Frames that can be sent from sender to receiver (all of which can contain data) are the following.

Single Frame (SF). Used only when the PDU is small enough to be carried inside a single FlexRay frame, that is,

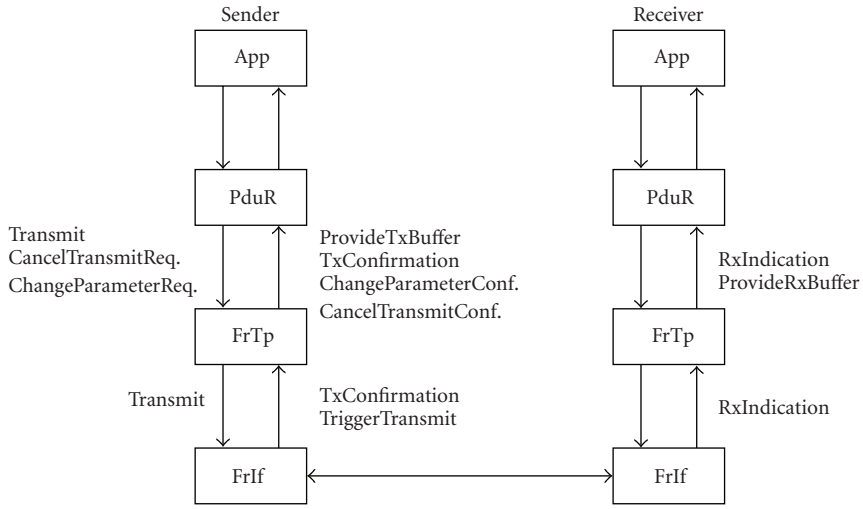

Figure 3: FrTp service primitives.

segmentation is not required. The header includes the length of the PDU.

First Frame (FF). When the PDU must be segmented, this is the first frame transmitted. The header includes the length of the PDU.

Consecutive Frame (CF). When the PDU must be segmented, all data, except that in the first frame, is sent in CFs. The header includes a sequence number to indicate the ordering of the CFs.

Frames that can be sent in the reverse direction (none of which contain data) are the following.

Flow Control (FC). Used to control the sending rate. Subtypes are: CTS to indicate the sender is clear to send; WAIT to indicate the sender must wait; and OVERFLOW to indicate the receiver has exhausted its buffer. The header includes the block size: the maximum number of CFs that are allowed to be transmitted before the next FC.

Acknowledgment (ACK). Used to acknowledge the current status of the PDU transfer. Subtypes are: positive (PACK) to indicate all data has been successfully received; and negative (NACK) to indicate an error (and either a retransmission should occur or the transfer should be aborted). The header includes sequence number field to indicate the faulty CF (in case of NACK).

An example of FrTp with segmented, acknowledged transfer with retries is illustrated in Figure 4. The FF is transmitted and upon receipt the receiver replies with a FC frame indicating the sender is clear to send the next frames. Included in the FC frame is the Block Size (BS) which indicates the maximum number of CFs the sender is allowed to send before waiting for the next Ack/CTS.

If a frame is lost or arrives with errors, the receiver sends a NACK indicating the frame that is missing (all frames contain a sequence number). Upon receiving the NACK the sender must retransmit all frames sent but not yet acknowledged. This is a go-back-N error/flow control scheme. Once all data is received by the receiver (the total data length is included in the First Frame), a PACK is sent indicating the successful completion of data transfer.

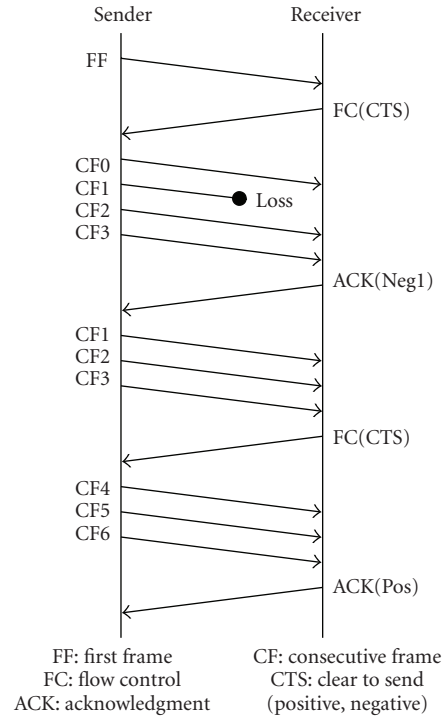

Figure 4: Example FrTp message sequence diagram.

4. Coloured Petri Nets

Coloured Petri nets [6] are a class of high-level nets that extend the features of basic Petri nets. The formal definition

of nonhierarchical CPNs (based on [6, Definition 4.3]) follows (In this paper hierarchical CPNs are used as they make the graphical net easier to read. The definition of hierarchical CPNs (see [6, Definition 6.1]) introduces three new net elements to Definition 1. However for simplicity we use the nonhierarchical definition as it is sufficient for explaining the model and analysis. In any case, a hierarchical CPN can be easily converted to a nonhierarchical CPN.). Definitions 1 to 6 are based on Definitions 4.2, 4.3, 4.4, 9.6, 9.19, and 9.9 from [6], respectively. Definition 7 is based on the definition in Section 2.3.2 of [31].

Definition 1. A nonhierarchical Coloured Petri Net is a nine-tuple CPN $= (P, T, A, \Sigma, V, C, G, E, I)$, where we find the following.

(1) P is a finite set of *places*.

(2) T is a finite set of *transitions* T such that $P \cap T = \emptyset$.

(3) $A \subseteq P \times T \cup T \times P$ is a set of directed *arcs*.

(4) Σ is a finite set of nonempty *colour sets*.

(5) V is a finite set of *typed variables* such that $\text{Type}[v] \in \Sigma$ for all variables for all $v \in V$.

(6) $C : P \to \Sigma$ is a *colour set function* that assigns a colour set to each place.

(7) $G : T \to \text{EXPR}_V$ is a *guard function* that assigns a guard to each transition t such that $\text{Type}[G(t)] =$ Boolean.

(8) $E : A \to \text{EXPR}_V$ is an *arc expression function* that assigns an arc expression to each arc a such that $\text{Type}[E(a)] = C(p)_{\text{MS}}$, where p is the place connected to the arc a.

(9) $I : P \to \text{EXPR}_\emptyset$ is an *initialisation function* that assigns an initialisation expression to each place p such that $\text{Type}[I(p)] = C(p)_{\text{MS}}$.

For the inscriptions (C, G, E, I), EXPR denotes the set of expressions provided by the inscription language CPN ML (an extension of Standard ML). The type of an expression e is denoted by $\text{Type}[e]$. MS refers to a multiset. Graphically, in this paper places are illustrated as eclipses, transitions as rectangles, guards in square brackets, and all other inscriptions located next to the corresponding place/arc.

The execution of a CPN consists of occurrence of transitions, or more precisely, *binding elements*. Concepts for the execution are defined in the following.

Definition 2. For a Coloured Petri Net CPN $= (P, T, A, \Sigma, V, C, G, E, I)$, the following concepts are defined.

(1) A *marking* is a function M that maps each place $p \in P$ into a multiset of *tokens* $M(p) \in C(p)_{\text{MS}}$.

(2) The *variables of a transition* t are denoted $\text{Var}(t) \subseteq V$ and consist of the free variables appearing in the guard of t and in the arc expressions of arcs connected to t.

(3) A *binding* of a transition t is a function b that maps each variable $v \in \text{Var}(t)$ into a value $b(v) \in \text{Type}[v]$. The set of all bindings for a transition is denoted $B(t)$.

(4) A *binding element* is a pair (t, b) such that $t \in T$ and $b \in B(t)$. The set of all binding elements $\text{BE}(t)$ for a transition t is defined by $\text{BE}(t) = \{(t, b) \mid b \in B(t)\}$. The set of all binding elements in a CPN model is denoted BE.

Definition 3. A binding element $(t, b) \in \text{BE}$ is *enabled* in a marking M if and only if the following two properties are satisfied:

(1) $G(t)\langle b \rangle$.

(2) For all $p \in P : E(p, t)\langle b \rangle \ll= M(p)$. When (t, b) is enabled in M, it may *occur*, leading to the marking M' defined by:

(3) For all $p \in P : M'(p) = (M(p) - -E(p, t)\langle b \rangle) + +E(t, p)\langle b \rangle$.

The symbols $++$, $--$ and $\ll=$ are multiset operations.

Definition 3 shows that the occurrence of a binding element while in marking M_1 produces a new marking M_2, or $M_1 \overset{(t,b)}{\to} M_2$. M_2 is said to be reachable from M_1 and the set of all markings reachable from M is $\mathcal{R}(M)$. By considering steps and occurrence sequences (see [6] for details) the state space of a CPN can be obtained as the set of all markings (states) reachable from the initial marking M_0. More precisely, we find the following.

Definition 4. The *state space* of a Coloured Petri net is a directed graph $\text{SS} = (N_{\text{SS}}, A_{\text{SS}})$ with arc labels from BE, where:

(1) $N_{\text{SS}} = \mathcal{R}(M_0)$ is the set of *nodes* or states.

(2) $A_{\text{SS}} = \{(M, (t, b), M') \in N_{\text{SS}} \times \text{BE} \times N_{\text{SS}} \mid M \overset{(t,b)}{\to} M'\}$ is the set of *arcs*.

Knowing all possible states, it is possible to prove properties such as deadlocks and livelocks of the system modelled by the CPN. Two properties that are considered in this paper relate to terminal markings and bounds.

Definition 5. Let a transition $t \in T$ and a marking M be given. M is a *terminal marking* if and only if for all $t \in T : \neg(M \overset{t}{\to})$.

Definition 6. Let a place $p \in P$ and a nonnegative integer $n \in \mathbb{N}$ be given. n is an *upper integer bound* for p if and only if for all $M \in \mathcal{R}(M_0) : |M(p)| \leq n$.

A state space can also be treated as a finite state automata (FSA) where the binding elements represent the alphabet accepted by the FSA. Theorems and algorithms developed for the analysis of FSAs can be applied to determine if the sequences of one state space/FSA is preserved in another state space/FSA. This approach is based on the fact that any nondeterministic FSA with ϵ-transitions, defined in

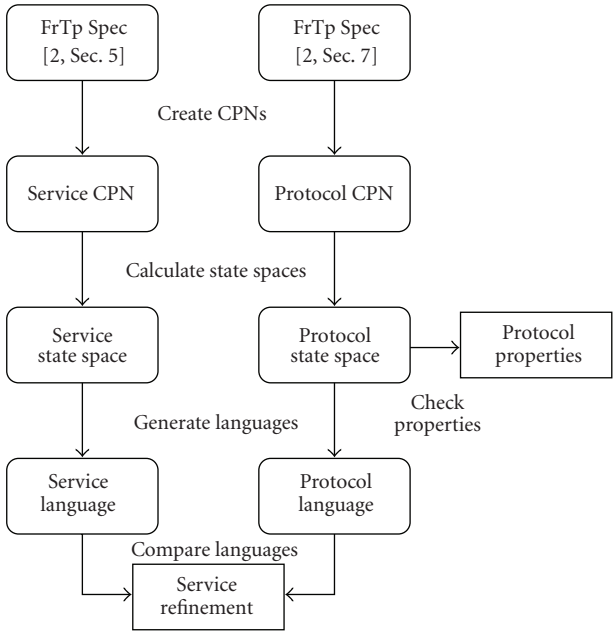

Figure 5: Protocol Verification Methodology for FrTp.

Definition 7, can be converted into a canonical form, a minimised deterministic FSA [31]. If the canonical forms are isomorphic then the two initial FSAs have the same language. Hence state space analysis is key to proving dynamic properties, as well as equivalence of the protocol and service language in protocol verification.

Definition 7. A nondeterministic FSA is a 5-tuple NDFSA $= (Q, \Sigma, s, H, \Delta)$ where, we find the following.

(1) Q is a finite set of states.

(2) Σ is an input alphabet.

(3) s is one of the states in Q designated as a start state.

(4) H is a set of halt or final states.

(5) Δ is a relation on $(Q \times \Sigma) \times Q$ and is called the transition relation.

5. CPN Model of FlexRay Transport Protocol

Protocol verification involves proving a protocol holds desired dynamic properties, such as absence of deadlocks, as well as proving that the protocol faithfully implements the desired service specification. The steps for achieving this are part of a commonly applied protocol engineering methodology [12]. Although different methods can be used, Coloured Petri nets are well suited to the task, as has been demonstrated in numerous examples [29, 32–34]. The steps applied in this paper are summarised in Figure 5.

Using the FrTp specification in [4], a CPN model of the FrTp protocol is created (see the right-side of Figure 5). Generating the state spaces for different initial configurations of the CPN allows specific protocol properties, such as absence of deadlock, to be proved. Section 5.2 describes the protocol CPN model, while Section 6 reports results from the state space analysis.

To show that the protocol specification faithfully refines the service specification, the protocol language (which can be obtained from the protocol state space) is compared to the service language. Although [4] lists the set of service primitives that can be exchanged between FrTp and the upper layer, as shown in Figure 3, there is no explicit definition of the possible orderings of service primitives. Therefore, based on our understanding of both the PDU Router (PduR) and FrTp protocol, we have developed a CPN model of the service that aims to capture the possible orderings of service primitives. The service language is obtained from the state space of the service CPN. Section 5.1 presents the service CPN model, while Section 6 describes how the languages are obtained from the state space and reports results from the language comparison.

Although not shown in Figure 5, the methodology is often used iteratively to verify a protocol in an incremental manner. For example, the verification may first be applied on a CPN that only models basic features, then applied again as optional features are introduced into the CPN.

5.1. FrTp Service CPN. The primary purpose of the FrTp service CPN, shown in Figure 6, is to generate the allowed possible sequences of service primitives, that is, the FrTp service language. The CPN model uses transitions to model the delivery of service primitives between FrTp and PduR (the transitions are highlighted with thick borders in Figure 6). Other transitions model local events/actions at the sender and receiver. As the interest is only in the sequences of service primitives, and such sequences should be independent of the protocol operations (e.g., retransmissions), details of the protocol operation are not modelled. The service language obtained from the service CPN is presented in Section 6.

5.2. FrTp Protocol CPN. To verify the functional properties of FrTp, a CPN model of the protocol has been created. This section outlines the design of the model, and presents part of the CPN.

5.2.1. Model Design Decisions. For the verification of FrTp to be successful, the FrTp CPN model must be a valid representation of the protocol specification and the analysis tools and techniques must be able to cope with the verification (e.g., avoid state space explosion). It is also desirable that the CPN is easy to maintain (e.g., to reflect potential changes in FrTp or add new features for performance analysis). To achieve this tradeoff between accuracy, clarity and details of the model, numerous design decisions were made. The key decisions and their justifications are described below.

Data Independence. As with many protocols, the operation of FrTp is independent of the actual data. That is, the actions of the sender and receiver are unaffected by the contents of the data fields in each frame. Therefore that actual data is not modelled in the CPN. However the operation of FrTp is dependent on the size of the data, for example, the sender

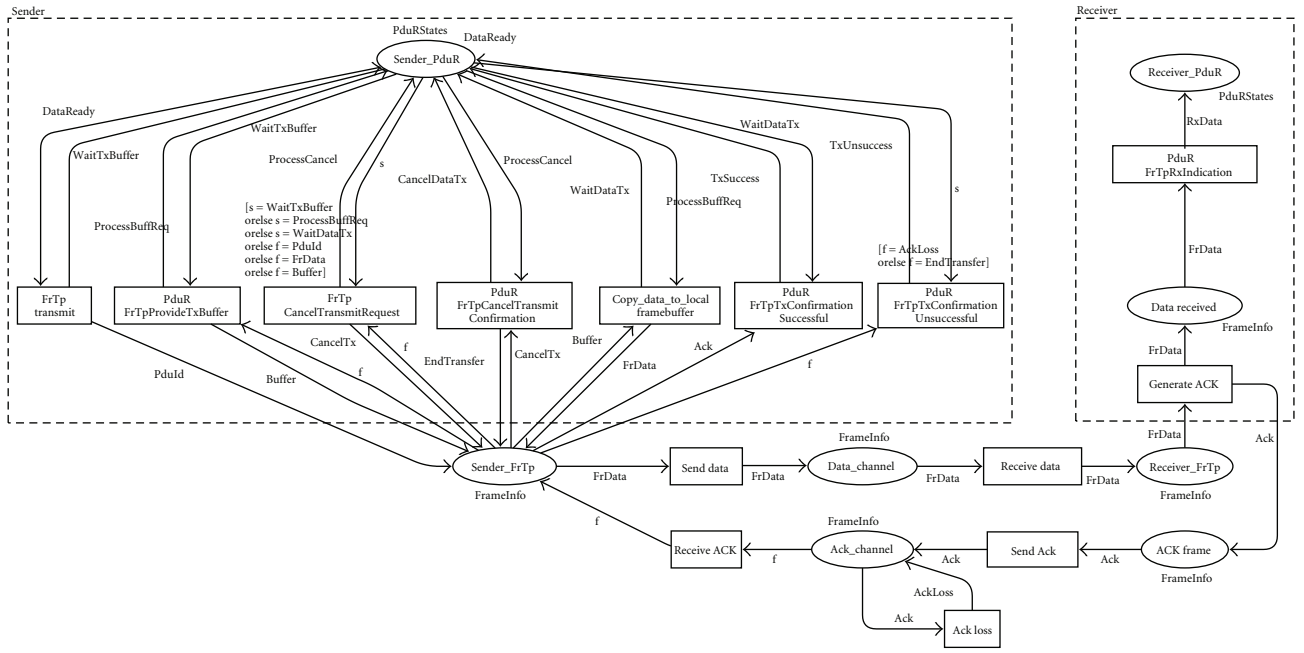

Figure 6: FrTp CPN Model: Service.

must decide whether a single frame can be sent carrying the whole data, or segmentation into multiple frames is needed. Therefore the CPN model considers the total size of the data and the maximum frame size (both in bytes) as inputs, and uses them to determine when to apply segmentation. An alternative design choice could have been to abstract from the actual size in bytes. That is, not explicitly modelling the total size or maximum frame size, but instead introducing a model parameter specifying the number of segments (1 to n). However this would not make state space analysis any easier and is potentially detrimental to the accuracy and clarity of the model.

Communication Channel. A FlexRay communication bus is full duplex, allowing multiple FlexRay controllers to communicate via a shared medium. Rather than model the detailed operations of the FlexRay data link and physical layer protocols, we assume an abstract channel model: a frame transmitted on the bus may either be received, in order, at the intended destination or the frame is lost (i.e., not received). This simplifies the CPN model and analysis, while also capturing the core behaviour of the channel from FrTp's perspective. For example, if an error occurs on the bus due to buffer overflows or link impairments, then a frame would not be received by the destination. From FrTp's perspective, no matter what caused the error, the frame was transmitted but not received, that is, lost. As the focus is only on functional properties of the protocol (not performance), modelling the reception of a frame in a nondeterministic manner allows our analysis to consider how FrTp behaves when errors occur in the FlexRay communication bus.

Number of PDUs. In AUTOSAR the PDU Router delivers data (referred to as a PDU) to FrTp, which then transmits the PDU as FrTp frames on the FlexRay communication bus. An important design decision is whether to model the exchange of an unlimited number of PDUs (which FrTp allows for) or only consider a finite number of PDUs, in particular a single PDU. The latter is preferable in order to simplify analysis, but only if the CPN still accurately reflects the core behaviour of FrTp. This will be true if the operation of FrTp when handling one PDU is unaffected by other PDUs.

For a single connection (from one sender to one receiver) we assume data PDUs from the PDU Router are handled sequentially by FrTp. That is, sender A must finish the transmission of the first PDU to receiver B before starting to transmit a second PDU to B. With overtaking not possible in the communication channel, the frames associated with the first PDU will always be delivered before the first frame associated with the next PDU is sent. In most cases, this means the operation of FrTp for different PDUs will be independent. The exception is when frames are delayed in the communication channel and the delivery of the PDU by the sender has been unsuccessful. Illustrated with a simple example in Figure 7, a delayed ACK associated with the first PDU can be misinterpreted as an ACK for the second PDU. This is a well-known problem in any acknowledgment-based protocol [11]. It can be solved by making the assumption a frame has a maximum delay in the communication channel and by limiting the transmission rate of PDUs such that a delayed ACK will be received before the next PDU is transmitted. We assume such a mechanism is in place (If this assumption was not made, then when modelling only the functional behaviour of FrTp an "error" would always be detected. However in practice, if the timing aspects of FrTp are well designed (e.g., timeout periods, transmission rates), this "error" will not occur. Hence we are faced with the modelling decision: include time in the model, which

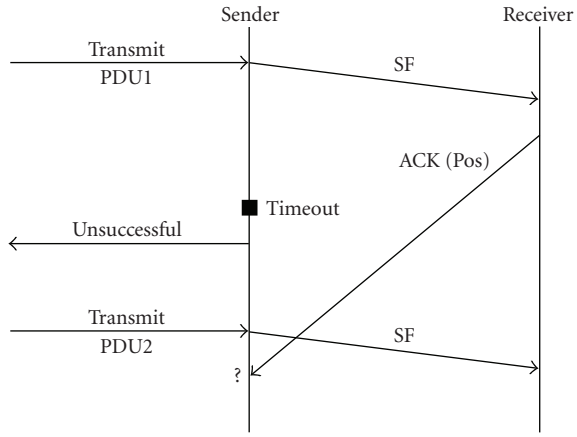

Figure 7: Example error with delayed ACK in FrTp.

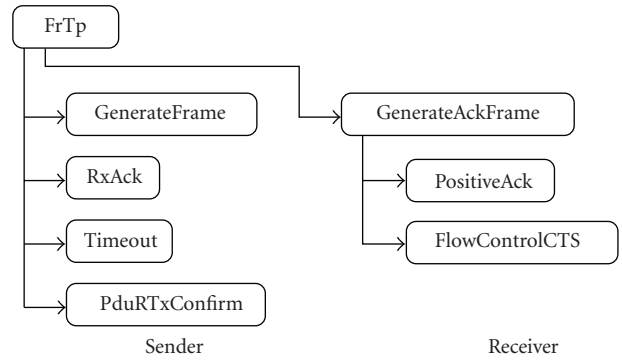

Figure 8: FrTp CPN model: hierarchy.

would add significant complexity, or assume the error will not occur. We chose the latter. Analysis of the timing issues is left for future work.). As a result, we assume sequential PDUs from one sender to one receiver are independent.

There is no assumption about sending in order to other receivers, that is, *A* may be transmitting a PDU to *B* while also transmitting another PDU to *C*. However in this case, as FrTp frames contain the address of the sender and receiver, even if *C* received a frame intended to *B* the frame would be disregarded. Hence it is assumed the operation of FrTp on one connection (a pair of sender/receiver) is unaffected by the operation of FrTp on another connection.

Finally PDU transmissions in opposite directions are assumed to be independent. With source/destination addresses included in frames and independent frame types in each direction (SF, FF and CF from sender; FC and ACK from receiver), frames arriving at *B* (and associated with a PDU sent from *A* to *B*) will not affect the way *B* operates when it is sending a PDU to *A*.

In summary, for most cases in FrTp the transmission of one PDU will not affect the transmission of another. However there is a specific case, delayed ACKs, when this is not true unless the sending rate is limited. We assume that the sending rate is limited, avoiding problems with delayed ACKs. Hence only a single PDU is considered in the CPN model and analysis.

Time. The FrTp sender uses a timer to determine when to retransmit a frame if an expected ACK has not yet been received. Time is not explicitly modelled in the CPN. Instead the event of a timeout is considered nondetermistic: if data has been sent and an ACK not yet received, then the timeout either may occur or it may not occur. This captures the possible behaviour of the protocol. Of course if the model is to be used for performance analysis in the future, explicit timing information would need to be added (which is possible with CPNs and the software CPN Tools).

Optional Features. The optional features of cancelling a transmission, changing parameters and overflow at the receiver are not modelled. In addition, only the basic ISO-compliant frame types are modelled (as opposed to the optional extended frames). Our decision to focus on the core features of FrTp first is part of the incremental approach to protocol verification. If the core features can be verified to be correct, then it may be possible to independently analyse the different options one-by-one (rather than analysing all features at once, which could quickly lead to the state space explosion problem arising). Modelling and analysing the optional features is left for future work.

5.2.2. Model Structure. FrTp is modelled as a hierarchical CPN, where there is a CPN on a single top-level page. The top-level page contains substitution transitions, which in fact represent a CPN on a subpage. The hierarchy is shown in Figure 8. The top-level page models the main steps of transmitting and receiving frames. Seven subpages model details of: generating data and ACK frames (separated as positive ACK and flow control frames at the receiver); processing an ACK upon reception; handling timeouts; and generating confirmation primitives to the PduR at the sender. In total there are 15 places and 26 transitions. Important constants and colour sets used in the model are in Figure 9. To support manual validation of the model against the specification, the naming scheme for CPN elements follows that used in the specification. Key pages are described in Section 5.2.3—the remaining are presented in the appendix.

5.2.3. Model Description. The flow of data for a single message sent from sender to receiver is highlighted by the thick arcs in the top-level page (Figure 10). On the left of the page is the sender, in the middle is the communication channel, and the right is the receiver. At the top the interface between PDU Router and FrTp is modelled. The initial marking of place `From_Tx_PduR is 1‘Transmit`. This indicates the PDU Router has data ready to transmit. The occurrence of transition `FrTp_Transmit` models the delivery of that data to FrTp.

The place `Data_to_Send` contains a 5-tuple: bytes already sent; bytes already acknowledged; sequence number of last frame sent; sequence number of last frame acknowledged; and current block size. This information is used

```
(*Default values of system constant*)
val WholeDataC = ref 400;
val FrameSizeC = ref 100;
val BSC = ref 2;
val InitialSNC = ref 0;
val MaxRetryC = ref 1;
val TimeoutEnableC = ref true;
val LossEnableC = ref true;
(*Data types (colour sets) *)
colset FrameType = with SingleFrame | FirstFrame | ConsecutiveFrame |
                        FlowControl | AckFrame;
colset FrameState = with CTS | WT | OVFLW | FrameStateNotSet;
colset Ack = with Positive | Negative | AckNotSet;
colset Frame = record ft:FrameType *dl:DataLength *fs:FrameState *sn:SN *ack:Ack *
                      bs:BlockSize *data:Data;
colse  Frames = list Frame;
colset NSDU = with Transmit | Indication | Successful | Unsuccessful;
colset DataToSend = product SentData *AckedData *SentSN *AckedSN *CurrentBS;
colset BSControl = product CanSent *NextTxSN;
colset DataReceived = product DataLength *RxdFrame *NextRxSN;
(*colsets not listed are of type in*)
(*variable and function definitions are not show*)
```

Figure 9: Selected FrTp CPN declarations.

and updated by transitions modelling the generation and reception of frames.

Frames containing data are generated and stored in the transmit buffer (place `Sender_FrTp`). This, as well as the receive buffer (place `Receiver_FrTp`), is modelled as a FIFO queue. The type of frame generated (single frame, first frame, consecutive frame) depends on the data size, block size and previous frames generated. This is modelled in detail in the `GenerateFrame` subpage (the transition `GenerateFrame` is a substitution transition, as indicated by the double lines).

Using the standard CPN ML constructs for modelling FIFO queues, a list of frames is stored in place `Data_Channel`. Initially empty, the occurrence of transition `TxFrame` adds a frame to the tail of the list and `RxFrame` extracts the frame from the head of the list. The function `LossEnable()` returns true if a model input configuration variable is set to enable frame loss. In that case, transition `Frame_Loss` will be enabled whenever a frame is in `Data_Channel`, and its occurrence deletes a frame from the head of the list.

Upon receiving a data frame, the receiver processes the received frame and potentially generates an ACK or FC frame. The detailed procedure is modelled on the subpage `GenerateAckFrame`. The place `Data_Received` models the total number of bytes expected, the bytes received, and the next expected sequence number in a 3-tuple. Once the bytes received equals the bytes expected the data can be delivered to the receiver PDU Router. This is modelled by the `FrTp_Indication` transition, putting an `Indication` token in place `To_Rx_PduR`.

As data frames are processed by the receiver, the FC, NACK and PACK frames may be sent on the return Ack channel. Upon reception the sender processes the frame, as modelled on subpage `Rx_Ack` (Figure 11). If a FC frame is received, depending on the number of retries (place `Number_of_Retry`), new data frames may be generated and sent. If a PACK is received, then the data transfer is successful and via the `PduR_FrTpTxConfirmation` subpage, a *Confirmation(Successful)* service primitive is delivered to the PDU Router. If a FC(Overflow) is received, or the maximum number of retries has been reached, a *Confirmation(Unsuccessful)* primitive is delivered to the PDU Router.

Key input parameters to the model are the following.

WD (Whole Data): Size of the data sent by the PDU Router [bytes].

FS (Frame Size): Maximum size of payload in a frame [bytes].

BZ (Block Size): Number of consecutive frames allowed to be sent before flow control received.

MR (Max Retry): Maximum number of retries sender makes before aborting the transfer

Loss: If true then frame loss is possible in the channel; otherwise a reliable channel is assumed

Varying the values of these parameters allows for investigation of FrTp under different conditions.

6. Verification of FlexRay Transport Protocol

Simulation of the CPN (i.e., stepping through selected occurrence sequences) was used to validate the operation of the model. Then state space and language analysis was performed in order to prove the properties of FrTp. This section defines the desired properties and reports the final set of results from the analysis.

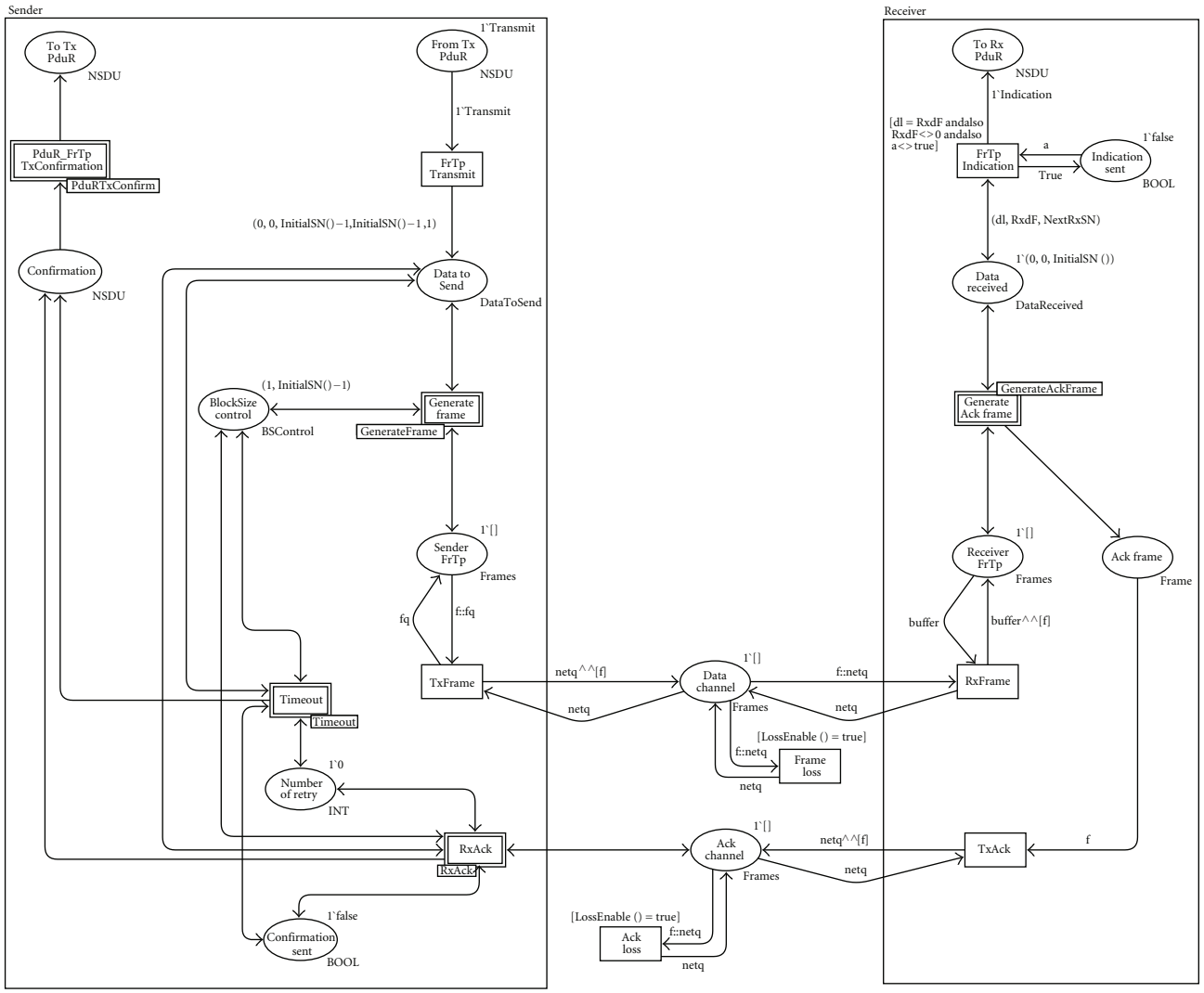

Figure 10: FrTp CPN Model: Top-level page.

6.1. Desired Properties. A key property of any protocol is absence of deadlocks. In this paper, deadlocks are defined as unexpected terminal markings in the state space, where the set of terminals markings is denoted as M_{TM}. We have designed the CPN model so that after a PDU is delivered, the sender/receiver do not revert back to an initial marking. Instead we consider those markings that the sender/receiver terminate in as *expected*. Three sets of expected terminal markings—M_{SS}, M_{UU} and M_{US}—are defined based on the intended operation of FrTp. For brevity assume:

$$P_{\mathrm{channel}} = \{\texttt{Data\_Channel}, \texttt{Ack\_Channel}\},$$
$$P_{\mathrm{buffer}} = \{\texttt{Sender\_FrTp}, \texttt{Receiver\_FrTp}\}. \tag{1}$$

Marking M_{SS} should be reached if both sender and receiver have successfully completed the data transfer, that is, *Confirmation(Successful)* primitive delivered to sender PduR and *Indication* delivered to receiver PduR:

$$M_{\mathrm{SS}}(p) = \begin{cases} \text{Successful} & \text{if } p = \texttt{To\_Tx\_PduR} \\ (0, 0, x, x, \_) & \text{if } p = \texttt{Data\_to\_Send} \\ \text{Indication} & \text{if } p = \texttt{To\_Rx\_PduR} \\ \text{true} & \text{if } p = \texttt{Indication\_Sent} \\ (WD, WD, \_) & \text{if } p = \texttt{Data\_Received} \\ [\,] & \text{if } p \in \mathrm{P}_{\mathrm{channel}} \\ [\,] & \text{if } p \in \mathrm{P}_{\mathrm{buffer}} \\ \varnothing_{\mathrm{MS}} & \text{if } p = \texttt{Confirmation} \\ \varnothing_{\mathrm{MS}} & \text{if } p = \texttt{Ack\_Frame} \\ \_ & \text{otherwise.} \end{cases} \tag{2}$$

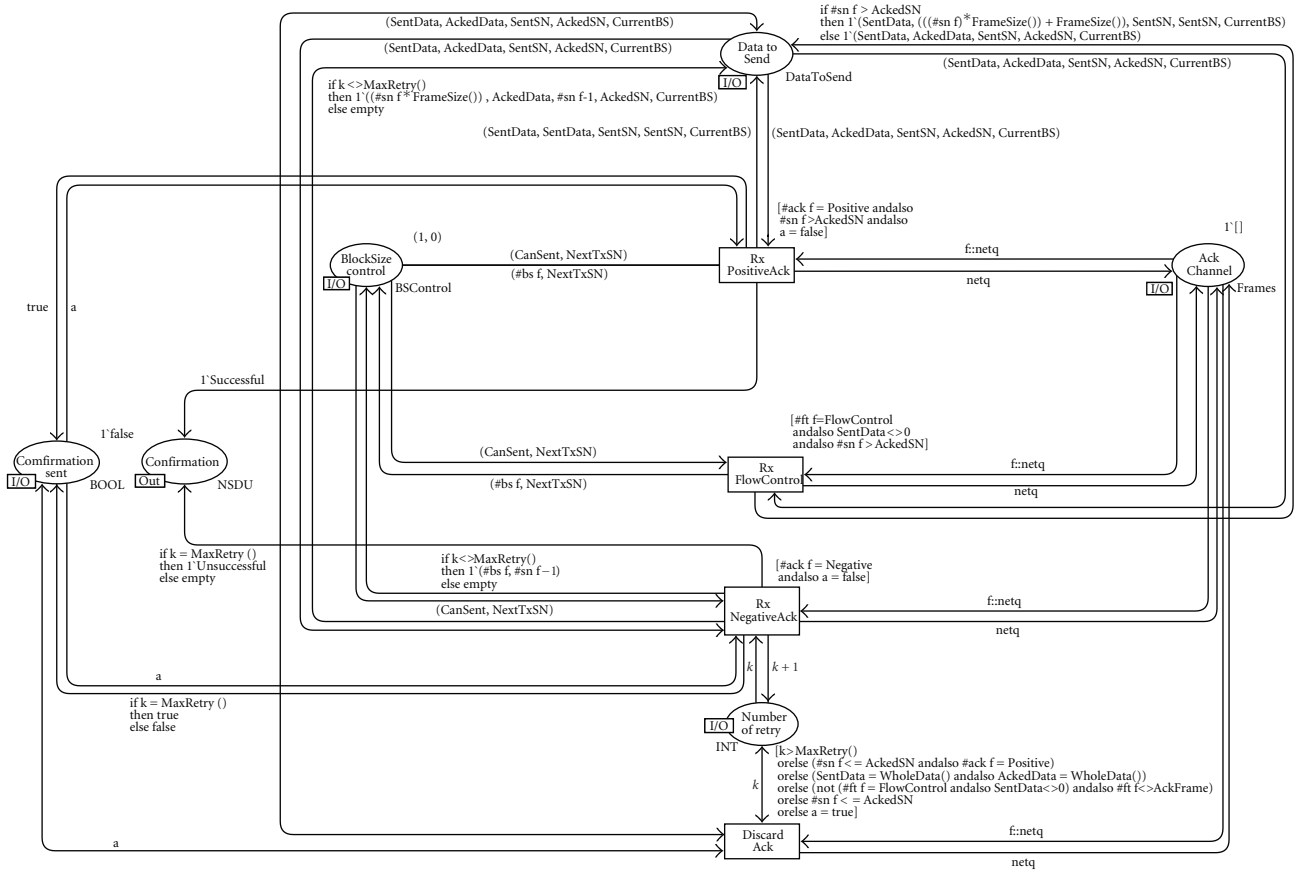

Figure 11: FrTp CPN Model: RxAck.

$\varnothing_{\text{MS}}$ is the empty multiset (no tokens in the place), and following Standard ML notation _ is any value and [] is an empty list.

Marking M_{UU} should be reached if the data transfer is unsuccessful and both transmitter and receiver are aware of the failure (*Confirmation (unsuccessful)* at sender PduR and *Indication* at receiver PduR):

$$M_{\text{UU}}(p) = \begin{cases} \text{Unsuccessful} & \text{if } p = \texttt{To\_Tx\_PduR} \\ \varnothing_{\text{MS}} & \text{if } p = \texttt{To\_Rx\_PduR} \\ \text{false} & \text{if } p = \texttt{Indication\_Sent} \\ [] & \text{if } p \in \text{P}_{\text{channel}} \\ [] & \text{if } p \in \text{P}_{\text{buffer}} \\ \varnothing_{\text{MS}} & \text{if } p = \texttt{Confirmation} \\ \varnothing_{\text{MS}} & \text{if } p = \texttt{Ack\_Frame} \\ \_ & \text{otherwise.} \end{cases} \tag{3}$$

Finally, there may be a case where the receiver successfully receives the data (*Indication* delivered to receiver PduR), but the transmitter is not informed of this (e.g., the acknowledgment cannot be delivered successfully), and hence *Confirmation(Unsuccessful)* delivered to sender PduR:

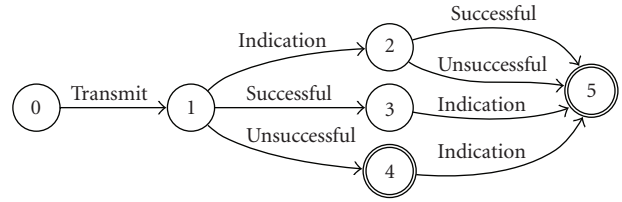

Figure 12: Desired FrTp service language.

$$M_{\text{US}}(p) = \begin{cases} \text{Unsuccessful} & \text{if } p = \texttt{To\_Tx\_PduR} \\ \text{Indication} & \text{if } p = \texttt{To\_Rx\_PduR} \\ \text{true} & \text{if } p = \texttt{Indication\_Sent} \\ (WD, WD, \_) & \text{if } p = \texttt{Data\_Received} \\ [] & \text{if } p \in \text{P}_{\text{channel}} \\ [] & \text{if } p \in \text{P}_{\text{buffer}} \\ \varnothing_{\text{MS}} & \text{if } p = \texttt{Confirmation} \\ \varnothing_{\text{MS}} & \text{if } p = \texttt{Ack\_Frame} \\ \_ & \text{otherwise.} \end{cases} \tag{4}$$

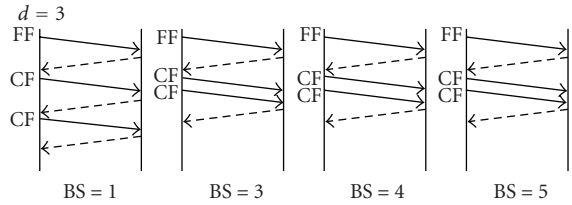

Figure 13: Impact of increasing BS greater than d.

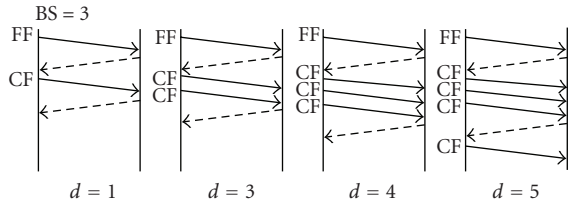

Figure 14: Impact of increasing d.

The first desired property of FrTp is the following.

Property 1 (Absence of deadlocks). The FrTp CPN contains no deadlocks if and only if:

$$M_{\mathrm{TM}} = M_{\mathrm{SS}} \cup M_{\mathrm{UU}} \cup M_{\mathrm{US}}. \tag{5}$$

Ideally the protocol language and desired service language shall be equivalent. The desired service language ($\mathcal{L}_S$) is defined as the language accepted by the FSA in Figure 12.

Property 2 (Language equivalence). The FrTp Protocol language ($\mathcal{L}_P$) is equivalent to the FrTp Service language ($\mathcal{L}_S$) if and only if

$$\mathcal{L}_P \subseteq \mathcal{L}_S \wedge \mathcal{L}_S \subseteq \mathcal{L}_P. \tag{6}$$

Understanding the number of frames that can be in the network at any one time is useful for dimensioning transmit/receive buffers as well as the FlexRay bus capacity and utilisation. Although not explicitly stated in the specification [4], both the buffers and channels shall be bounded. It is difficult to know the upper bounds prior to analysis, and hence state space analysis was used to determine the bounds. Therefore in this paper rather than defining the bounds as properties, we simply report the bounds measured from the state space analysis in the next section.

6.2. Analysis Approach. State space and language analysis has been applied to prove the desired properties of FrTp for selected sets of input parameter values, as well as make other observations on its behaviour. For brevity, the specific configurations of the FrTp protocol CPN (and corresponding state space) are referred to as d-BS-MR-Loss where d = WD/FS and Loss is T if loss is modelled, otherwise F.

6.2.1. State Space Analysis. For a given set of input parameter values, CPN Tools can calculate the full state space of the FrTp CPN. Then CPN ML queries can be applied to prove properties from the state space. For example, a CPN ML query function was written to check if all terminal markings are one of either M_{SS}, M_{UU} or M_{US}.

6.2.2. Language Analysis. The FrTp service CPN (Section 5.1) was used to generate a service state space in CPN Tools. The service state space is $\mathrm{SS}_S = (N_S, A_S)$ and has initial marking M_{0_S} and terminal markings TM_S. By treating the state space as a nondeterministic FSA, as defined in Definition 8, where binding elements for the transitions representing the service primitives of interest are mapped to symbols of the alphabet (recall that the optional features of changing parameters and cancelling transmissions are not considered), the service language $\mathcal{L}_S$ can be obtained. The language is shown in Figure 12.

Definition 8. The nondeterministic FSA of the FrTp service is $\mathrm{NDFSA}_S = (Q, \Sigma, s, H, \Delta)$ where $Q = N_S$, $s = M_{0_S}$, $\Sigma = \{\mathrm{Transmit}, \mathrm{RxIndication}, \mathrm{Successful}, \mathrm{Unsuccessful}, \epsilon\}$, $H = \mathrm{TM}_S$ and Δ can be obtained from arcs of the service state space, A_S, where a binding element (t, b) is mapped to symbol, l, in the alphabet as

$$l = \begin{cases} \mathrm{Transmit} & \text{if } (t,b) = (\texttt{FrTp\_Transmit}, \langle\rangle) \\ \mathrm{RxIndication} & \text{if } (t,b) = (\texttt{PduR\_FrTpRxIndication}, \langle\rangle) \\ \mathrm{Successful} & \text{if } (t,b) = (\texttt{PduR\_FrTpTxConfirmation\_Successful}, \langle\rangle) \\ \mathrm{Unsuccessful} & \text{if } (t,b) = (\texttt{PduR\_FrTpFrTpTxConfirmation\_UnSuccessful}, \langle f = \_, s = \_\rangle) \\ \epsilon & \text{otherwise.} \end{cases} \tag{7}$$

Similarly, each state space of the FrTp protocol CPN can be treated as a NDFSA, and the protocol language, $\mathcal{L}_P$, obtained. For brevity, the definition of NDFSA_P is not shown however it is similar to Definition 8 but the service primitives are mapped from the four transitions `FrTp_Transmit`, `FrTp_Indication`,

Successful and Unsuccessful (the last two, on page PduRTxConfirm).

6.3. Results. Results from the state space and language analysis have been collected for a range configurations. In particular, both the cases of no loss and loss have been considered. In this paper we present only results when frame loss is enabled. The case of no loss produced similar results and much smaller state spaces. In fact, because of the nondeterministic nature of the CPN, every occurrence sequence possible with no loss, is also possible when loss is enabled (i.e., the Frame_Loss and Ack_Loss transitions do not occur). In addition, we only present results when WD is an integer number of FS = 100.

6.3.1. No Retransmissions. State spaces have been calculated from the FrTp protocol CPN when no retransmissions are allowed (MR = 0) for the 100 combinations of configurations $[1\cdots10]$-$[1\cdots10]$-0-T. Results are presented in Table 1. The first column gives the configuration, and the next two columns the number of nodes and arcs in the state space. The "Terminals" column gives the total number of terminal markings, followed by a triple giving the count of M_{SS}, M_{UU} and M_{US}. The last column gives the upper integer bound on places Data_channel and Ack_channel.

Note that some results from the 100 configurations are not shown (although were obtained)—they will be discussed shortly.

The first observation from the state space analysis is that Property 1 (absence of deadlocks) is proved to hold for all configurations $[1\cdots10]$-$[1\cdots10]$-0-T (since $|M_{TM}| = |M_{SS}| + |M_{UU}| + |M_{US}|$).

For each configuration the FrTp protocol language was also obtained. Every protocol language produced was compared to the service language in Figure 12, and all were equivalent hence proving Property 2 for configurations $[1\cdots10]$-$[1\cdots10]$-0-T.

State space analysis is powerful for investigating dynamic properties of the FrTp CPN. However there are two limitations of this analysis approach as can be readily observed from the results so far: the state explosion problem, where eventually the state space becomes too large to calculate in reasonable time/memory and the dependence on initial parameter values, therefore calculating the state space for every possible combination is time consuming or even impossible (The state space sizes reported in Table 1 are manageable, as a state space with 140,000 nodes and 600,000 arcs takes about 90 minutes to calculate on a Intel Core 2 Quad Q8400 2.6 GHz processor with 8 GB memory.). Properties 1 and 2 are proven for only for configurations $[1\cdots10]$-$[1\cdots10]$-0-T. However by observing trends in the state spaces generated from these configurations, we can gain increased confidence that the properties should hold for other configurations.

First note that for the configurations with BS $\geq d - 1$, the state space of the protocol is independent of BS. For example, the state spaces of configurations 3-2-0-T, 3-3-0-T and 3-4-0-T are identical (whereas 3-1-0-T is different) (Although all results for configurations $[1\cdots10]$-$[1\cdots10]$-0-T were calculated, this is why some results are omitted from Table 1.). This is because, after the first frame (FF), there are no more than BS CF frames to transmit, resulting in the same protocol behaviour independent of the value of BS. This is illustrated with an example in Figure 13.

Proposition 1. *For the FrTp protocol CPN: MR = 0, Loss = T, for all* $d \in \{1,\ldots,10\}$ *for all* $BS \in \mathbb{N}^* \mid BS \geq d - 1$: *Properties 1 and 2 hold.*

Proof. Proof is by inspection of the FrTp protocol CPN. BS is used only when the receiver generates an ACK (modelled by transitions on page GenerateAckFrame and its subpages). The bs field in an ACK is set to the value of BS by the functions FLOWCONTROL(), PACK() and NACK(). When the sender receives an ACK (transitions on RxAck page), the value of the bs field is stored in the place BlockSize_Control. We denote M(BlockSize_Control) = (b, sn) where b counts the blocks allowed to be sent and sn is the next sequence number to use. The value of b is only modified by the following sets of transitions occurring.

(1) RxPositiveAck, RxFlowControl and RxNegativeAck on page RxAck set b to the value of bs in the received ACK frame.

(2) Timeout sets b to the value of bs received in the previous ACK (taken from place Data_to_Send).

(3) All transitions that create data frames, T_{generate} = {GenerateFirstFrame, GenerateSingleFrame, GenerateConsecutiveFrame}, decrement b by 1.

Only the occurrence of the transitions T_{generate} depend on the value of b. That is, for all $t \in T_{\text{generate}}$: $G(t)$ =IsXFrame$(\cdots)\wedge$Valid$(\cdots)$ where X refers to the type of frame (first, single or consecutive). A condition for Valid() to be true is $b \neq 0$. Also note that transitions T_{generate} increment the SentData whenever they occur (stored in place Data_to_Send). A second condition for Valid() to be true is SentData < WD. Therefore transitions T_{generate} both decrement the count of blocks to be sent (b) and increment the data already sent (SentData). The guards on these transitions prevent them from occurring if either the maximum number of frames allowed by BS have been sent or all d frames has been sent. If GenerateFirstFrame has occurred once, then there are $d - 1$ consecutive frames left to be sent. If BS $\geq d - 1$ then after sending all $d - 1$ consecutive frames then $SentData$ = WD and for all $t \in T_{\text{generate}}$: $G(t)$ = false. As no transitions other than T_{generate} depend on b (and hence BS), increasing BS will not change the set of transitions that can occur. Therefore, the same state space will be generated, irrespective of BS. As full state spaces have been calculated, and Properties 1 and 2 proven true for the configurations $[1\cdots10]$-$[1\cdots10]$-0-T, Proposition 1 holds. □

A second observation from the state spaces of configurations $[1\cdots10]$-$[1\cdots10]$-0-T is that their sizes (number of

Table 1: State space results for FrTp protocol CPN with no retries (1).

Config	Nodes	Arcs	Terminals	Bounds
1-1-0-T	38	67	4 (1,2,1)	(1,1)
2-1-0-T	62	105	7 (1,5,1)	(1,1)
3-1-0-T	86	143	10 (1,8,1)	(1,1)
4-1-0-T	110	181	13 (1,11,1)	(1,1)
5-1-0-T	134	219	16 (1,14,1)	(1,1)
6-1-0-T	158	257	19 (1,17,1)	(1,1)
7-1-0-T	182	295	22 (1,20,1)	(1,1)
8-1-0-T	206	333	25 (1,23,1)	(1,1)
9-1-0-T	230	371	28 (1,26,1)	(1,1)
10-1-0-T	254	409	31 (1,29,1)	(1,1)
1-2-0-T	38	67	4 (1,2,1)	(1,1)
2-2-0-T	62	105	7 (1,5,1)	(1,1)
3-2-0-T	118	219	11 (1,9,1)	(2,1)
4-2-0-T	142	257	14 (1,12,1)	(2,1)
5-2-0-T	198	371	18 (1,16,1)	(2,1)
6-2-0-T	222	409	21 (1,19,1)	(2,1)
7-2-0-T	278	523	25 (1,23,1)	(2,1)
8-2-0-T	302	561	28 (1,26,1)	(2,1)
9-2-0-T	358	675	32 (1,30,1)	(2,1)
10-2-0-T	382	713	35 (1,33,1)	(2,1)
1-3-0-T	38	67	4 (1,2,1)	(1,1)
2-3-0-T	62	105	7 (1,5,1)	(1,1)
3-3-0-T	118	219	11 (1,9,1)	(2,1)
4-3-0-T	295	665	16 (1,14,1)	(3,2)
5-3-0-T	319	703	19 (1,17,1)	(3,2)
6-3-0-T	375	817	23 (1,21,1)	(3,2)
7-3-0-T	552	1263	28 (1,26,1)	(3,2)
8-3-0-T	576	1301	31 (1,29,1)	(3,2)
9-3-0-T	632	1415	35 (1,33,1)	(3,2)
10-3-0-T	809	1861	40 (1,38,1)	(3,2)
1-4-0-T	38	67	4 (1,2,1)	(1,1)
2-4-0-T	62	105	7 (1,5,1)	(1,1)
3-4-0-T	118	219	11 (1,9,1)	(2,1)
4-4-0-T	295	665	16 (1,14,1)	(3,2)
5-4-0-T	793	2143	22 (1,20,1)	(4,3)
6-4-0-T	817	2181	25 (1,23,1)	(4,3)
7-4-0-T	873	2295	29 (1,27,1)	(4,3)
8-4-0-T	1050	2741	34 (1,32,1)	(4,3)
9-4-0-T	1548	4219	40 (1,38,1)	(4,3)
10-4-0-T	1572	4257	43 (1,41,1)	(4,3)
6-5-0-T	2075	6453	29 (1,27,1)	(5,4)
7-5-0-T	2099	6491	32 (1,30,1)	(5,4)
8-5-0-T	2155	6605	36 (1,34,1)	(5,4)
9-5-0-T	2332	7051	41 (1,39,1)	(5,4)
10-5-0-T	2830	8529	47 (1,45,1)	(5,4)
7-6-0-T	5158	17850	37 (1,35,1)	(6,5)
8-6-0-T	5182	17888	40 (1,38,1)	(6,5)
9-6-0-T	5238	18002	44 (1,42,1)	(6,5)
10-6-0-T	5415	18448	49 (1,47,1)	(6,5)

Table 1: Continued.

Config	Nodes	Arcs	Terminals	Bounds
8-7-0-T	12200	45837	46 (1,44,1)	(7,6)
9-7-0-T	12224	45875	49 (1,47,1)	(7,6)
10-7-0-T	12280	45989	53 (1,51,1)	(7,6)
9-8-0-T	27679	110844	56 (1,54,1)	(8,7)
10-8-0-T	27703	110882	59 (1,57,1)	(8,7)
10-9-0-T	60756	255689	67 (1,65,1)	(9,8)

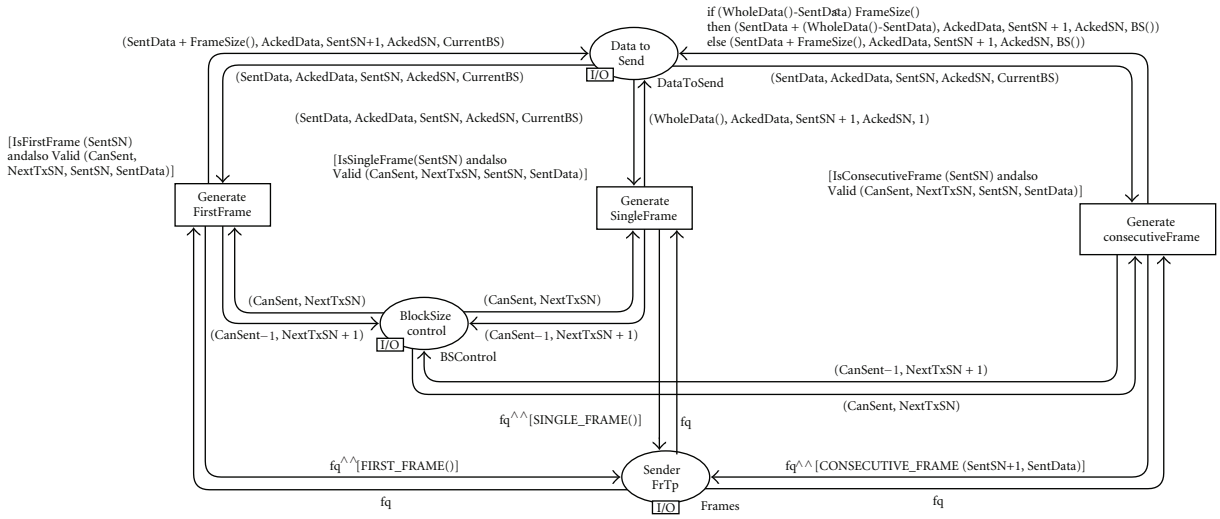

Figure 15: FrTp CPN model: GenerateFrame.

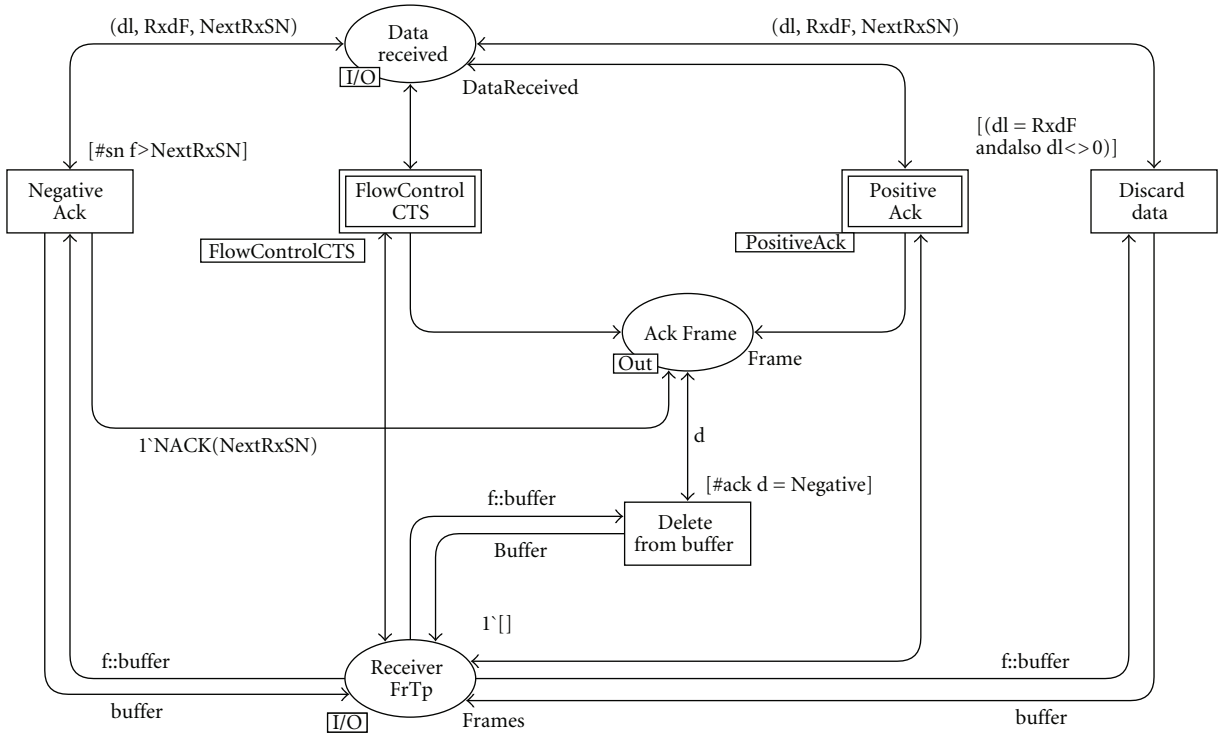

Figure 16: FrTp CPN model: GenerateAckFrame.

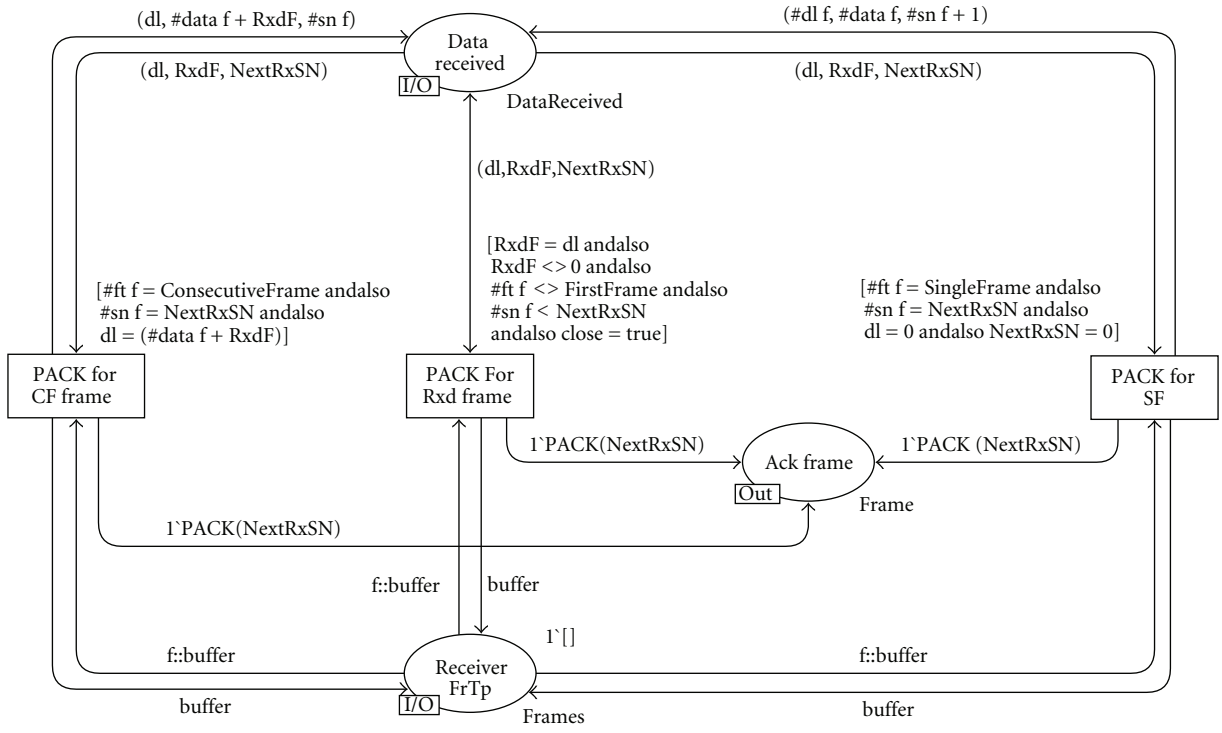

Figure 17: FrTp CPN model: PositiveAck.

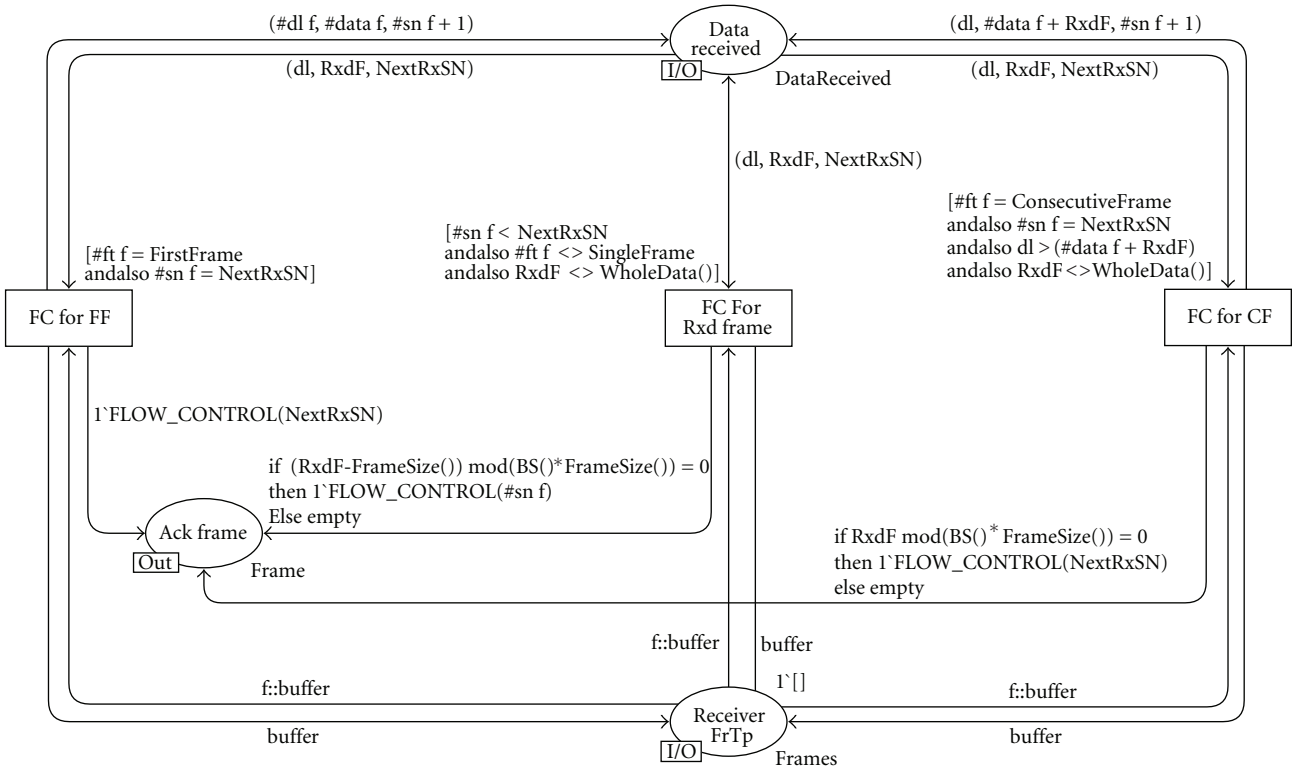

Figure 18: FrTp CPN model: FlowControl CTS.

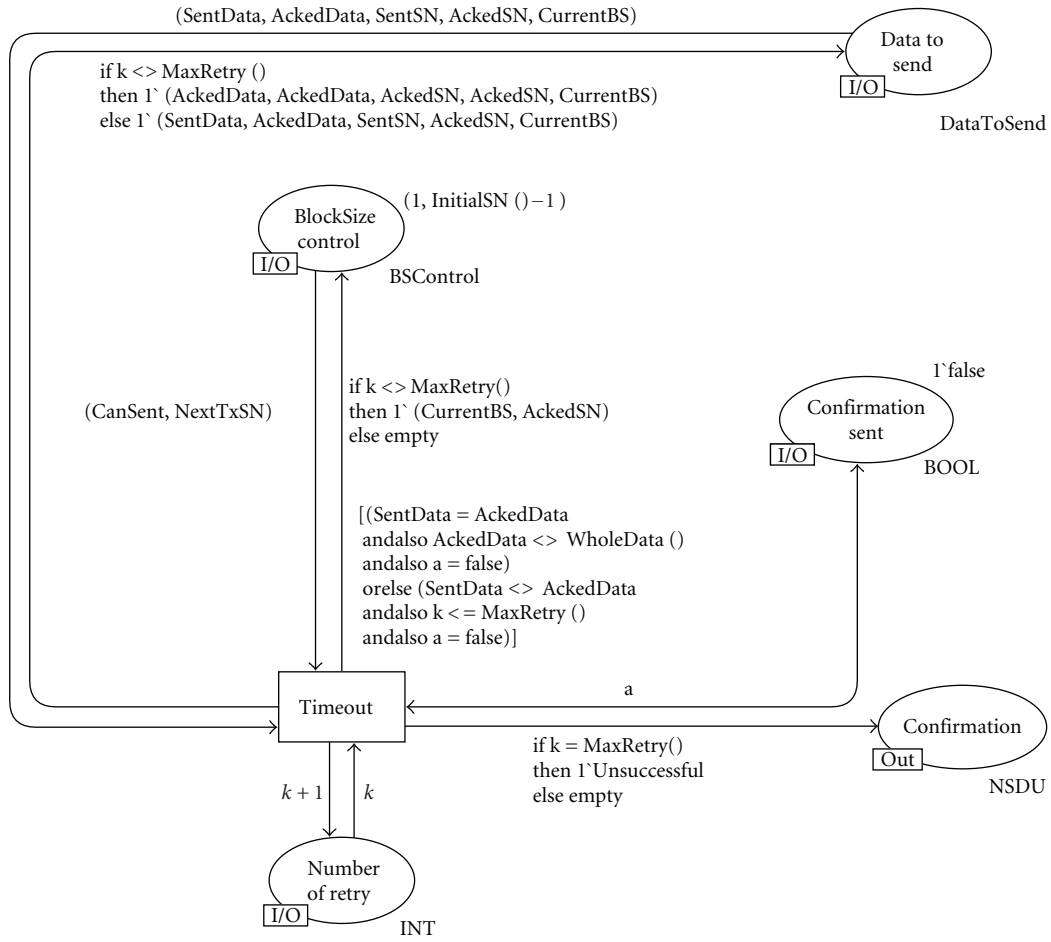

Figure 19: FrTp CPN model: timeout.

nodes, $|N|$, and arcs, $|A|$) are directly related to d and BS. By applying curve fitting, the number of nodes in the state space of the configurations analysed can be expressed as

$$|N_{d\text{-BS-0-T}}| = |N_{1\text{-BS-0-T}}| + \left(\left\lfloor \frac{d + \text{BS} - 2}{\text{BS}} \right\rfloor \sum_{i=1}^{\text{BS}} c_i - \sum_{i=X}^{\text{BS}} c_i \right),$$

$$\text{where } X = \text{BS} - \text{BS} \left\lfloor \frac{d + \text{BS} - 2}{\text{BS}} \right\rfloor + d,$$

$$|N_{1\text{-BS-0-T}}| = 38,$$

$$\mathbf{c} = [24, 56, 177, 498, 1282, 3083, 7042, 15479, 33077]. \quad (8)$$

An identical formula is obtained for the number of arcs, $|A_{d\text{-BS-0-T}}|$, except $|A_{1\text{-BS-0-T}}| = 67$ and $\mathbf{c} = [38, 114, 446, 1478, 4310, 11397, 27987, 65007, 144845]$.

Intuitively, for a fixed BS as d increases, there are more frames to transmit leading to a larger state space. In the cases when the sender must wait for an ACK before proceeding the protocol operations are sequential. For example, from the results in Table 1 when BS = 1, incrementing d results in an additional consecutive frame and ACK, thereby an additive increase in the state space size. However with larger values of BS, increasing d allows for different levels of concurrency among transitions in the CPN. For example consider Figure 14. For $d = 2$ there is only one CF, however with $d = 3$, two CFs are sent, and an increased state space results because of both the extra frame as well as the possible interleavings between the transitions modelling transfer of the two CFs. With $d = 4$ there are even more possible interleavings with three CFs, leading it an even larger increase in state space size.

The significance of finding a closed form solution for the state space size is threefold: by analysing selected configurations, increased confidence that the protocol operates correctly is gained for configurations not analysed; it suggests the CPN and protocol exhibit structure that could be utilised in further analysis to alleviate the state explosion problem (e.g., using state space reduction techniques) and to avoid the dependence on initial parameter values (i.e., parametric verification [35]); and it provides guidance as to which configurations can be analysed within reasonable time using CPN Tools.

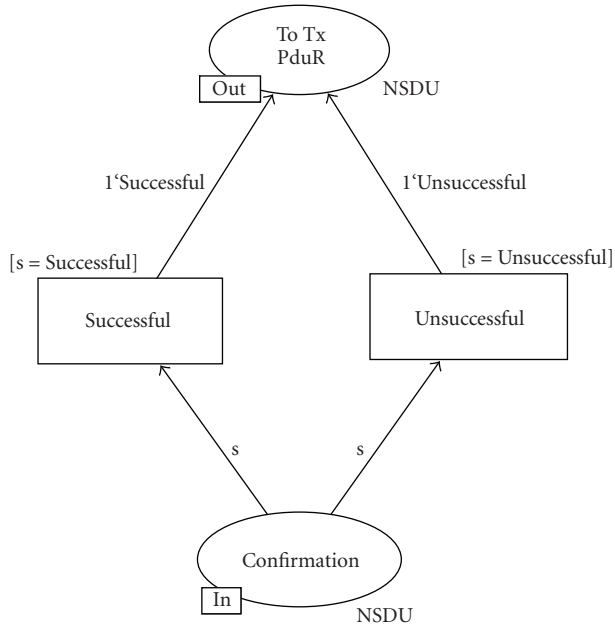

Figure 20: FrTp CPN model: PduRFrTpTx Confirmation.

If (8) (and the corresponding equation for arcs) holds for $d > 10$, BS > 10, then we conjecture that the desired properties will also hold.

Conjecture 2. *For the FrTp protocol CPN: MR* $= 0$*, Loss* $= T$*, for all* $d \in \mathbb{N}^*$*, for all BS* $\in \mathbb{N}^*$*: Properties 1 and 2 hold.*

Selected configurations have been tested and for each, the state space size matches that expected and the desired properties are true (see Table 2).

Using a similar approach as with the size of the state space, we can observe trends in the upper integer bounds of the channels. For configurations $[1\cdots10]$-$[1\cdots10]$-0-T the bounds can be expressed as:

$$UB_{\text{data}} = \begin{cases} 1 & \text{if } d = 0, \\ d-1 & \text{if } 1 < d \le \text{BS}, \\ \text{BS} & \text{if } d > BS, \end{cases}$$
$$UB_{\text{ack}} = \begin{cases} 1 & \text{if } d \le 2, \\ d-2 & \text{if } 2 < d \le \text{BS}, \\ \text{BS}-1 & \text{if } d > BS. \end{cases} \tag{9}$$

For the data channel, the sender cannot transmit more than BS frames before waiting for an ACK, and hence the bound is BS (unless the total number of frames to be transmitted, d, is less than BS). For the ack channel, the bound is BS $-$ 1 because in the case of the first CF lost, then for the remaining BS $-$ 1 CFs sent, the receiver will respond with a NACK, that is, BS $-$ 1 NACKs in the ack channel. We conjecture that this will be true for all configurations.

Conjecture 3. *For the FrTp protocol CPN: MR* $= 0$*, Loss* $= T$*, for all* $d \in \mathbb{N}^*$*, for all BS* $\in \mathbb{N}^*$*:* (9) *hold for the channel bounds.*

6.3.2. Retransmissions Allowed. With retransmissions allowed (MR > 0) the FrTp sender may retransmit a frame after a timeout occurs. Hence more concurrent operations are possible, leading to increased number of occurrence sequences (which increases as MR increases) as illustrated in the selected state space results in Tables 3 and 4.

Table 3 shows the results from the configuration 1-1-$[1\cdots10]$-T. There is only a single frame transmitted (or retransmitted). From the state space and language analysis for all 10 configurations Properties 1 and 2 hold. The state space size can be expressed as:

$$|N_{\text{1-1-MR-T}}| = \frac{1}{24}\sum_{i=0}^{4} c_i \text{MR}^i \tag{10}$$
$$\text{where } \mathbf{c} = [912, 1884, 1258, 336, 26].$$

An identical formula is obtained for the number of arcs, $|A_{1-1-\text{MR}-\text{T}}|$, except $\mathbf{c} = [1608, 4062, 3813, 1566159]$. For the number of terminal markings:

$$|T_{\text{1-1-MR-T}}| = 2(\text{MR} + 2) + 1. \tag{11}$$

For $d > 1$ the state space size grows rapidly for even small values of MR. Currently there are no closed form equations for the state space size in these cases.

Table 4 shows the results from selecting configurations with increasing d. Again, for all configurations Properties 1 and 2 hold.

For configurations $[1\cdots10]$-1-[1,2]-T-closed-form solutions for the state space size and number of terminals exist—for the number of nodes, see (12) and (13). Table 5 shows example state space results with larger variable values—the results match (10), (12) and (13), respectively,

$$|N_{d\text{-1-1-T}}| = \begin{cases} 184 & d = 1, \\ \frac{1}{2}\sum_{i=0}^{2} c_i (d-2)^i & d \ge 2, \end{cases} \tag{12}$$
$$\text{where } \mathbf{c} = [796, 521, 51]$$

$$|N_{d\text{-1-2-T}}| = \begin{cases} 534 & d = 1, \\ 1573 & d = 2, \\ \frac{1}{6}\sum_{i=0}^{3} c_i (d-3)^i & d \ge 3, \end{cases} \tag{13}$$
$$\text{where } \mathbf{c} = [20634, 13797, 2487, 138].$$

6.4. Discussion and Future Work. Coloured Petri nets and state space analysis has been used to prove the FrTp protocol, under selected configurations, is deadlock-free and conforms to the service specification when transferring

Table 2: State space results for FrTp protocol CPN with no retries (2).

Config	Nodes	Arcs	Terminals	Bounds
20-1-0-T	494	789	61 (1,59,1)	(1,1)
20-2-0-T	782	1473	70 (1,68,1)	(2,1)
20-3-0-T	1604	3693	79 (1,77,1)	(3,2)
20-4-0-T	3315	8969	88 (1,86,1)	(4,3)
20-5-0-T	6904	21301	97 (1,95,1)	(5,4)
20-6-0-T	15422	53454	106 (1,104,1)	(6,5)
20-7-0-T	26399	97993	113 (1,111,1)	(7,6)
20-8-0-T	55577	222219	120 (1,118,1)	(8,7)
11-10-0-T	130006	568613	79 (1,77,1)	(10,9)

Table 3: State space results for FrTp protocol CPN with no retries (1).

Config	Nodes	Arcs	Terminals	Bounds
1-1-1-T	184	467	7 (3,2,2)	(2,1)
1-1-2-T	534	1669	9 (5,2,2)	(3,1)
1-1-3-T	1211	4303	11 (7,2,2)	(4,1)
1-1-4-T	2364	9158	13 (9,2,2)	(5,1)
1-1-5-T	4168	17182	15 (11,2,2)	(6,1)
1-1-6-T	6824	29482	17 (13,2,2)	(7,1)
1-1-7-T	10559	47324	19 (15,2,2)	(8,1)
1-1-8-T	15626	72133	21 (17,2,2)	(9,1)
1-1-9-T	22304	105493	23 (19,2,2)	(10,1)
1-1-10-T	30898	149147	25 (21,2,2)	(11,1)

a single PDU from sender to receiver. Although there are several ambiguities in the text, no significant errors have been identified in the protocol specification in [4]. For a protocol to be used for in-vehicle communication, applying formal methods to gain a higher degree of confidence in the correct operation is still beneficial. However there are still issues to be considered in future work.

Model Expressiveness. The use of CPNs represents a tradeoff between model expressiveness and analysis capabilities. A focus of the model development was to create accurate specifications of the protocol that could be understood by nonCPN-experts and used in other protocol engineering tasks. Creating graphical models using a structure, naming schemes and constructs that are similar to those used in the specification is one way to aid the validation of the model. Other approaches to validate the model should also be considered. The model expressiveness however comes at the expense of formal analysis capabilities: other methods/tools (e.g., SPIN) can analyse much larger state spaces.

FrTp Features Not Analysed. Several optional features of the FrTp specification were not modelled or analysed. The CPN could be extended to include these features, and then analysed again. However its desirable to take advantage of the existing results and avoid the limitations of state space analysis. For example, component-based modelling and analysis techniques may be applicable for features such as changing parameters during a FrTp connection.

In the model design, assumptions were made about the independence of multiple PDUs. Ideally, a formal proof of independence or formal analysis with the assumptions relaxed is required for complete verification of FrTp. However as the issue with delayed ACKs can be solved with timing constraints, more practical insights into FrTp could be obtained by integrating time into the CPN and conducting performance analysis.

Limitations of State Space Analysis. In the paper the limitations of state space analysis of FrTp quickly become apparent. For instance, FrTp supports block sizes of 1 to 16: with retransmissions possible only block sizes of 1 to 4 (and small values of d) are analysed. To overcome these limitations, by analysing trends in the state space results and closer inspection of the CPN, observations on the desired properties are made without generating the complete set of state spaces. Proposition 1 shows, under certain conditions, the desired properties hold for all block sizes greater than $d-1$. Several conjectures that the properties hold are presented. The selected state space analysis provides increased confidence that FrTp is error-free. Proof of the conjectures, which requires detailed manual analysis of the CPN model using for example techniques applied in [29], is left for future work.

Another promising technique to alleviate the state explosion problem is to utilise the sweep-line method [36]. This method discards states from memory if they can no longer be reached. This has been applied for other protocols, where

Table 4: State space results for FrTp protocol CPN with retries (2).

Config	Nodes	Arcs	Terminals	Bounds
1-1-1-T	184	467	7 (3,2,2)	(2,1)
2-1-1-T	398	1037	11 (3,6,2)	(2,2)
3-1-1-T	684	1793	17 (3,12,2)	(2,2)
4-1-1-T	1021	2699	23 (3,18,2)	(2,2)
5-1-1-T	1409	3755	29 (3,24,2)	(2,2)
6-1-1-T	1848	4961	35 (3,30,2)	(2,2)
7-1-1-T	2338	6317	41 (3,36,2)	(2,2)
8-1-1-T	2879	7823	47 (3,42,2)	(2,2)
9-1-1-T	3471	9479	53 (3,48,2)	(2,2)
10-1-1-T	4114	11285	59 (3,54,2)	(2,2)
1-1-2-T	534	1669	9 (5,2,2)	(3,1)
2-1-2-T	1573	5254	13 (5,6,2)	(3,3)
3-1-2-T	3439	11644	22 (5,15,2)	(3,3)
4-1-2-T	6176	21127	33 (5,26,2)	(3,3)
5-1-2-T	9880	34119	44 (5,37,2)	(3,3)
6-1-2-T	14689	51128	55 (5,48,2)	(3,3)
7-1-2-T	20741	72662	66 (5,59,2)	(3,3)
8-1-2-T	28174	99229	77 (5,70,2)	(3,3)
9-1-2-T	37126	131337	88 (5,81,2)	(3,3)
10-1-2-T	47735	169494	99 (5,92,2)	(3,3)
1-1-3-T	1211	4303	11 (7,2,2)	(4,1)
2-1-3-T	4747	18608	15 (7,6,2)	(4,4)
3-1-3-T	13112	52597	24 (7,15,2)	(4,4)
4-1-3-T	28486	115521	35 (7,26,2)	(4,4)
5-1-3-T	53587	218659	46 (7,37,2)	(4,4)
6-1-3-T	91792	375751	57 (7,48,2)	(4,4)
7-1-3-T	146961	602453	68 (7,59,2)	(4,4)

Table 5: State space results for FrTp protocol CPN with retries (3).

Config	Nodes	Arcs	Terminals	Bounds
1-1-15-T	115103	593962	35 (31,2,2)	(16,1)
50-1-1-T	71654	206525	299 (3,294,2)	(2,2)
14-1-2-T	109501	392772	143 (5,136,2)	(3,3)

properties can be proved for state spaces 2-3 times the size of when sweep-line is not applied [34]. In FrTp the number of retries could be used as a progress measure to determine which states can/cannot be reached in the future.

7. Conclusions

AUTOSAR is an architecture for developing and deploying embedded applications in vehicles. One of the many components of AUTOSAR is the FlexRay Transport Protocol, which provides additional reliability and efficiency to the FlexRay inter-ECU communication bus. This paper has presented a formal Coloured Petri net model of both the FrTp protocol and service specifications. Using formal analysis techniques, the models enabled the verification of FrTp for selected configurations. This includes proving the absence of deadlocks, conformance of the protocol to the service specification, and characterisation of the upper bounds of buffers when a single-protocol data unit is transferred from FrTp sender to receiver. Proof of these properties, and further insights relating the state space to protocol parameters, provides a high-degree of confidence that the design of FlexRay Transport Protocol is error-free.

Appendix

FrTp Protocol CPN

Figures 15, 16, 17, 18, 19, and 20 illustrate the remaining pages of the FrTp protocol CPN model.

Acknowledgment

The authors would like to thank the anonymous reviewers for their many helpful suggestions for improving this paper.

References

[1] J. A. Cook, I. V. Kolmanovsky, D. McNamara, E. C. Nelson, and K. V. Prasad, "Control, computing and communications: technologies for the twenty-first century model T," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 334–355, 2007.

[2] FlexRay Consortium. FlexRay Protocol Specification. V2.1 Rev A, http://www.flexray.com/.

[3] AUTOSAR, "Technical Overview. V2.2.2 R3.1," 2008, http://www.autosar.org/.

[4] AUTOSAR, "Specification of FlexRay Transport Layer. V2.2.2 R3.1," 2008, http://www.autosar.org/.

[5] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded system design for automotive applications," *Computer*, vol. 40, no. 10, pp. 42–51, 2007.

[6] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer, Berlin, Germany, 2009.

[7] S. Choosang, R. Taburan, and S. Gordo, "A formal model of an AUTOSAR in-vehicle communications protocol," in *Proceedings of International Conference on Information and Communication Technology for Embedded Systems*, Bangkok, Thailand, 2010.

[8] OSEK/VDX. OSEK Communications Specification. V3.0.3, http://www.osek-vdx.org/.

[9] ISO, "Road Vehicles—Diagnostics on Controller Area Networks (CAN)—Part 2: network layer services," ISO 15765-2, 2004.

[10] F. Babich and L. Deotto, "Formal methods for specification and analysis of communication protocols," *IEEE Communications Surveys and Tutorials*, vol. 4, no. 1, pp. 2–20, 2002.

[11] G. J. Holzmann, *Design and Validation of Computer Protocols*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1991.

[12] J. Billington, G. E. Gallasch, and B. Han, "A Coloured Petri net approach to protocol verification," in *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, pp. 210–290, Springer, 2004.

[13] Department of Computer Science, University of Aarhus. CPN Tools, http://wiki.daimi.au.dk/cpntools/cpntools.wiki.

[14] M. Di Natale and A. L. Sangiovanni-Vincentelli, "Moving from federated to integrated architectures in automotive: the role of standards, methods and tools," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, 2010.

[15] J.-L. Boulanger and V. Q. Dao, "Requirements engineering in a model-based methodology for embedded automotive software," in *Proceedings of the IEEE International Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*, pp. 263–268, Ho Chi Minh City, Vietnam, 2008.

[16] K. Klobedanz, C. Kuznik, A. Thuy, and W. Mueller, "Timing modeling and analysis for AUTOSAR-based software development—a case study," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition*, Dresden, Germany, 2010.

[17] D. Bertrand, S. Faucou, and Y. Trinquet, "An analysis of the AUTOSAR OS timing protection mechanism," in *Proceedings of the IEEE Conference on Emerging Technologies Factory Automation*, pp. 1–8, Mallorca, Spain, 2009.

[18] T. Nolte, I. Shin, M. Behnam, and M. Sjodin, "A synchronization protocol for temporal isolation of software components in vehicular systems," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 4, pp. 375–387, 2009.

[19] S. Piao, H. Jo, S. Jin, and W. Jung, "Design and implementation of RTE generator for automotive embedded software," in *Proceedings of the 7th ACIS International Conference on Software Engineering Research, Management and Applications*, pp. 159–165, Haikou, China, 2009.

[20] D. Schreiner, M. Schordan, and K. Goschka, "Component based middleware-synthesis for AUTOSAR basic software," in *Proceedings of the IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, pp. 236–243, Tokyo, Japan, 2009.

[21] J. Grossmann, D. Serbanescu, and I. Schieferdecker, "Testing embedded real time systems with TTCN-3," in *Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation*, pp. 81–91, Denver, Colo, USA, 2009.

[22] G. Park, D. Ku, S. Lee, W.-J. Won, and W. Jung, "Test methods of the AUTOSAR application software components," in *Proceedings of the ICROS-SICE International Joint Conference*, pp. 2601–2606, Fukuoka, Japan, 2009.

[23] R. Kaivola, "Using compositional predorders in the verification of sliding window protocol," in *Proceedings of the 9th International Conference on Computer Aided Verification*, vol. 1254 of *Lecture Notes in Computer Science*, pp. 48–59, Springer, Haifa, Israe, 1997.

[24] M. A. Smith and N. Klarlund, "Verification of a sliding window protocol using IOA and MONA," in *Proceedings of the IFIP Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification*, pp. 19–34, Pisa, Italy, 2000.

[25] B. Badban, W. Fokkink, J. F. Groote, J. Pang, and J. Van De Pol, "Verification of a sliding window protocol in μcRL and PVS," *Formal Aspects of Computing*, vol. 17, no. 3, pp. 342–388, 2005.

[26] M. Chechik and H. Wang, "Bisimulation analysis of SDL-expressed protocols: a case study," in *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*, pp. 1–13, Mississauga, Canada, 2000.

[27] P. Abdulla, A. Annichini, and A. Bouajjani, "Symbolic verification of lossy channel systems: application to the bounded retransmission protocol," in *Proceedings of 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, vol. 1579, pp. 208–222, Springer, Amsterdam, The Netherlands, 1999.

[28] A. Valmari and I. Kokkarinen, "Unbounded verification results by finite-state compositional techniques: 10^{any} states and beyond," in *Proceedings of International Conference on Application of Concurrency to System Design*, pp. 75–85, Fukushima, Japan, 1998.

[29] G. E. Gallasch and J. Billington, "Parametric language analysis of the class of stop-and-wait protocols," in *Proceedings of the 29th International Conference on the Application and Theory of Petri Nets and Other Models of Concurrency*, Xi'an, China, 2008.

[30] O. Kallel, S. Dridi, and S. Hasnaoui, "Modeling and evaluating a CAN controller components using stochastic and colored petri nets," *International Review on Computers and Software*, vol. 4, no. 1, pp. 142–151, 2009.

[31] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Berlin, Germany, 2nd edition, 2001.

[32] S. Vanit-Anunchai, "Towards formal modelling and analysis of SCTP connection management," in *Proceedings of the Ninth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus, Denmark, 2008.

[33] L. Liu and J. Billington, "Verification of the capability exchange signalling protocol," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3-4, pp. 305–326, 2007.

[34] S. Gordon, L. M. Kristensen, and J. Billington, "Verification of a revised WAP wireless transaction protocol," in *Proceedings of the 23st International Conference on Application and Theory of Petri Nets*, vol. 2360 of *Lecture Notes in Computer Science*, pp. 182–202, Springer, Adelaide, Australia, 2002.

[35] G. E. Gallasch and J. Billington, "A parametric state space for the analysis of the infinite class of stop-and-wait protocols," in *Proceedings of the 13th International SPIN Workshop on Model Checking of Software*, pp. 201–218, Vienna, Austria, 2006.

[36] S. Christensen, L. M. Kristensen, and T. Mailund, "A sweep-line method for state space exploration," in *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2031 of *Lecture Notes in Computer Science*, pp. 450–464, Springer, Genova, Italy, 2001.