

An Improved Architectural Specification of the Internet Open Trading Protocol*

Chun Ouyang, Lars Michael Kristensen**, and Jonathan Billington

Computer Systems Engineering Centre
School of Electrical and Information Engineering
University of South Australia
Mawson Lakes Campus, SA 5095, AUSTRALIA
chuoy001@students.unisa.edu.au {lars.kristensen,jonathan.billington}@unisa.edu.au

Abstract. The *Internet Open Trading Protocol* (IOTP) is an international standard (RFC 2801) currently being developed by the Internet Engineering Task Force. IOTP aims at providing an interoperable framework for electronic commerce (e-commerce) over the Internet. IOTP is expected to evolve into one of the central building blocks for the developing of next generation e-commerce on the Internet. We apply Coloured Petri Nets and Design/CPN for modelling and analysing IOTP, focussing on its protocol architecture. The contribution of this paper is the construction of a CPN specification of RFC 2801. Based on the constructed CPN model, we propose a complete and simplified architecture of IOTP compared with the partial architectural specification given in the RFC. We apply simulation, message sequence charts, and state space analysis to validate the CPN models of IOTP exchanges and transactions, and to validate that the suggested protocol architecture conforms to the specification of IOTP given in the RFC.

1 Introduction

Today's Internet is revolutionising commerce. It provides the first affordable and secure way to link people and computers spontaneously across organisational boundaries. Electronic commerce (e-commerce), traditionally conducted with the use of information technologies based on *Electronic Data Interchange* (EDI) over proprietary *Value Added Networks* (VAN), is rapidly moving onto the Internet [17]. With the advent of Internet-based e-commerce, such as online shopping and online retailing, a multitude of technology standards and specifications have been (or are being) developed to build the required communication infrastructures.

The *Internet Open Trading Protocol* (IOTP) [3,4] is a standard being developed by the Internet Engineering Task Force (IETF) [7], and is expected to evolve into one of the central building blocks for the developing of next generation e-commerce on the Internet. IOTP aims at providing an interoperable framework for e-commerce over the Internet. It is designed to specify the communication protocols to complete an electronic business transaction between two parties which have no prior association. The initial focus of IOTP is on the payment and delivery aspects of business-to-consumer (b-c) e-commerce.

Concurrently with the specification work being conducted within IETF, a number of research groups and companies are planning and working on the first trial implementations of selected parts of IOTP. This includes the project of the Open Trading Protocol toolkit for Java

* Supported by (1) Australian Technology Network (ATN) Small Research Grant, and (2) University of South Australia Divisional Small Grant.

** Supported by the Danish Natural Science Research Council.

(JOTP) in Xenosys Corporation [9], and the project of the Standard smart card Integrated settlement system (SMILE) [14] conducted by Hitachi [8]. There is however still no complete implementation of IOTP, and an interoperability test between independent implementations has not yet been conducted. As an emerging communication protocol, the development of IOTP is still in an early stage and can therefore benefit from the use of formal methods for modelling and analysis before becoming an Internet standard. Moreover, there are several aspects of IOTP that need development. First of all, IOTP lacks a detailed specification of its protocol architecture and as a consequence the internal organisation of IOTP needs clarification. Apart from that, RFC 2801 contains several ambiguities in the specification of the IOTP protocol itself.

In this paper we present our results from applying Coloured Petri Nets (CP-nets or CPNs) [10, 11] and the Design/CPN tool [13] for the modelling and analysis of IOTP. The primary focus of the work presented has been on the specification of the IOTP protocol architecture based on the modelling of the IOTP *baseline transactions*. The set of baseline transactions is what constitutes the main service provided by IOTP. The construction of the CPN models has identified the important protocol layers and components, and clarified their relationship.

This paper is organised as follows. Section 2 gives a brief overview and introduction to IOTP using the IOTP purchase transaction as an example. Section 3 gives an overview of the CPN model of IOTP. Sections 4 and 5 explain how the set of IOTP transactions and exchanges have been modelled. Section 6 presents the simulation and the state space analysis results. Section 7 presents the IOTP protocol architecture derived from the constructed CPN model. Finally, in Section 8 we summarize our contribution and discuss future work. The reader is assumed to be familiar with Coloured Petri Nets and Design/CPN.

2 The Internet Open Trading Protocol

In this section we introduce the basic concepts of IOTP. Baseline IOTP [3] defines eight transactions: *Authentication*, *Deposit*, *Withdrawal*, *Purchase*, *Refund*, *Value Exchange*, *Inquiry*, and *Ping*. The first six transactions are designed for business interactions and will be referred to as *trading transactions*¹. The last two transactions are independent of business processes and are so-called *infrastructure transactions*. Baseline IOTP trading transactions are consumer oriented. The Authentication transaction supports the authentication of a party involved in a transaction to validate that the party is who it claims to be. The Purchase transaction is carried out for the purchase of goods or services using some payment method. The Refund transaction occurs when the refund of a payment is required, usually as a result of an earlier purchase. The Deposit and the Withdrawal transactions are used for the deposit or the withdrawal of electronic cash at a financial institution. Finally, Value Exchange transaction supports the transfer or the conversion of an amount of electronic cash in one currency using one payment method to an amount of electronic cash in the same or another currency using the same or another payment method. The Inquiry transaction can be used to obtain information on the status of an ongoing or completed trading transaction, while the Ping transaction tests basic connectivity between parties involved in a trading transaction. Below we use the Purchase transaction to illustrate the basic operation of IOTP.

¹ In RFC 2801 [3], they are referred to as Authentication- and Payment-related IOTP Transactions (A-P Transactions).

2.1 IOTP Purchase Transaction

The Purchase transaction can be used when one party wants to purchase goods or services from another party over the Internet. Figure 1 shows a possible sequence of messages exchanged between the parties involved in a Purchase transaction. Each column of the Message Sequence Chart (MSC) corresponds to a so-called *trading role*. Trading roles are used to identify the different roles that organisations can play in a trade. IOTP defines five trading roles: *Consumer*, *Merchant*, *Payment Handler*, *Delivery Handler*, and *Merchant Customer Care Provider*. The *Consumer* is the organisation which receives and pays for the goods/services. The *Merchant* supplies the goods and receives payment for them. The *Payment Handler* is the organisation receiving the money from the *Consumer* on behalf of the *Merchant*. The *Delivery Handler* is the organisation responsible for delivering the goods to the *Consumer* on behalf of the *Merchant*. Finally, the *Merchant Customer Care Provider* (which we will not consider further in this paper) is the organisation responsible for resolving *Consumer* disputes and problems on behalf of the *Merchant*.

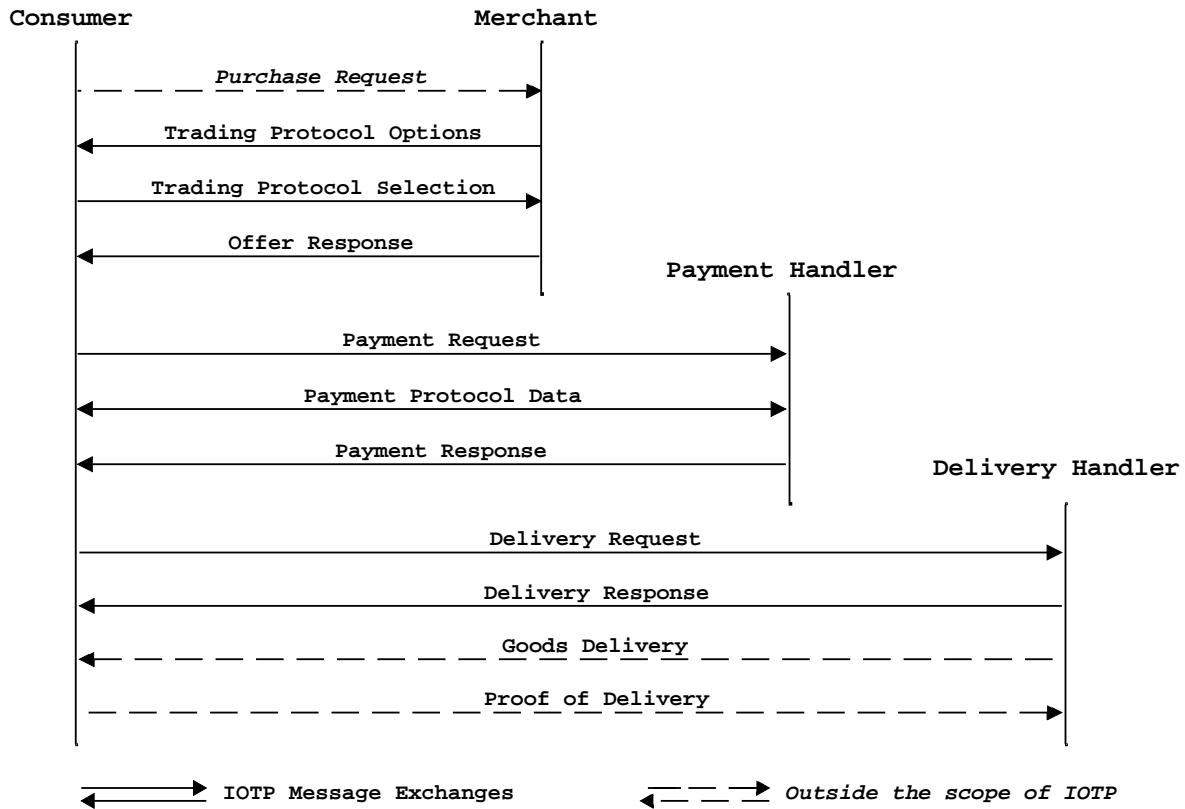


Fig. 1: A possible message flow in a Purchase transaction.

In the following we consider in detail the messages exchanged between trading roles in a Purchase transaction as shown in Fig. 1. The first phase in the Purchase transaction is the negotiation between the Consumer and Merchant regarding the payment brand and the payment protocol to be used. At some point the Consumer decides to buy goods from the Merchant, and sends a Purchase Request to the Merchant by, e.g., clicking on a button in a

web browser. Whereas this is outside the scope of IOTP, it enables the start of the Purchase transaction at the other side, i.e., the Merchant side. The Merchant then starts the Purchase transaction upon receiving the Purchase Request from the Consumer. The Merchant offers a list of Trading Protocol Options to the Consumer. The Trading Protocol Options include a list of payment brands (e.g., Visa Credit, MasterCard, Mondex card) that are accepted by the Merchant and the correspondent payment protocols available (e.g., SET², Mondex VTP³). The Consumer selects the payment brand (e.g. MasterCard) and the payment protocol (e.g. SET) among the options offered by the Merchant, and sends them back to the Merchant in a Trading Protocol Selection. The Merchant uses the selection made by the Consumer and the related information to create the Offer Response, and sends it to the Consumer. The Offer Response contains details of the goods, pay amount, and delivery instructions. More generally, the Offer Response can be considered as an invoice before the actual payment is carried out.

The next phase in the transaction is the payment. After checking the payment information contained in the Offer Response, the Consumer sends a Payment Request to the Payment Handler. The Payment Handler checks the information provided (such as a signature) in the Payment Request. If the information is valid, the payment is carried out via the exchange of Payment Protocol Data between the Consumer and the Payment Handler. This data could be SET protocol data if the SET protocol was selected as the payment protocol. When the payment exchanges finish, the Payment Handler sends a Payment Response which includes the payment receipt and an optional signature. This provides the Consumer with proof of the payment.

The final phase in the transaction is the delivery of the goods. The Consumer checks the delivery instructions in the Offer Response, and uses the payment receipt as authorisation in the Delivery Request that is sent to the Delivery Handler. The Delivery Handler starts or schedules the delivery, and sends a delivery note in the Delivery Response to the Consumer. The delivery of the goods might be a physical delivery, or an on-line delivery if the goods are electronic such as an electronic journal. Finally, the Consumer sends Proof of Delivery to the Delivery Handler upon receiving the goods. It should be mentioned that the last two steps in the phase of delivery, i.e., the actual delivery of goods/service and the proof of delivery, are outside the scope of IOTP.

From the above it can be seen that a Purchase transaction is conducted in three phases. At first, the choice of available payment brand and payment protocol is negotiated between Consumer and Merchant. Secondly, the payment is made between Consumer and Payment Handler. Thirdly, the delivery occurs between Consumer and Delivery Handler. A Purchase transaction can therefore be seen as being divided into three sub-transactions. IOTP defines two sets of such sub-transactions called *trading exchanges* and *document exchanges*. Each document and trading exchange involves a set of messages exchanged between trading roles. All eight IOTP baseline transactions can be expressed as combinations of such document and trading exchanges. Document exchanges are constructed from trading exchanges by grouping together parts of trading exchanges. One of the contributions of this paper is to merge the set of trading and document exchanges into one common set which we call *IOTP exchanges* leading to a simpler specification and architecture of IOTP.

² Secure Electronic Transaction (SET) is an open technical standard developed by Visa and MasterCard to facilitate secure payment card transactions on the Internet [16].

³ Mondex Value Transfer Protocol (VTP) enables a secure and legitimate transfer between two Mondex cards [15].

3 Overview of CPN IOTP Model

A CPN model has been constructed for each of the six trading transactions of IOTP. All six CPN models have a similar structure and share a common set of pages. All the CPN models could in principle be integrated into one single CPN model capturing all trading transactions. This has however not yet been done. We therefore focus on the CPN model of the Purchase transaction. The CPN models of the other trading transactions are similar. Since the Purchase transaction involves all types of IOTP exchanges, it can be considered a representative example.

Figure 2 shows the hierarchy page of the CPN model. The prime page Purchase provides the most abstract view of the Purchase transaction and has five subpages. Four of these subpages: Consumer, Merchant, PHandler, and DHandler correspond to the four trading roles involved in a Purchase transaction, and they specify how a Purchase transaction is implemented for each of the trading roles. We will refer to these pages as trading role pages. The prime page and the trading role pages constitute the transaction layer in the protocol architecture of IOTP which we will specify in Sect. 7. The specification of the transactions are presented in detail in Sect. 4. The last subpage of Purchase is Transport. It models the transport medium over which the trading roles communicate. In this paper we will not go into detail with the modelling of the transport layer.

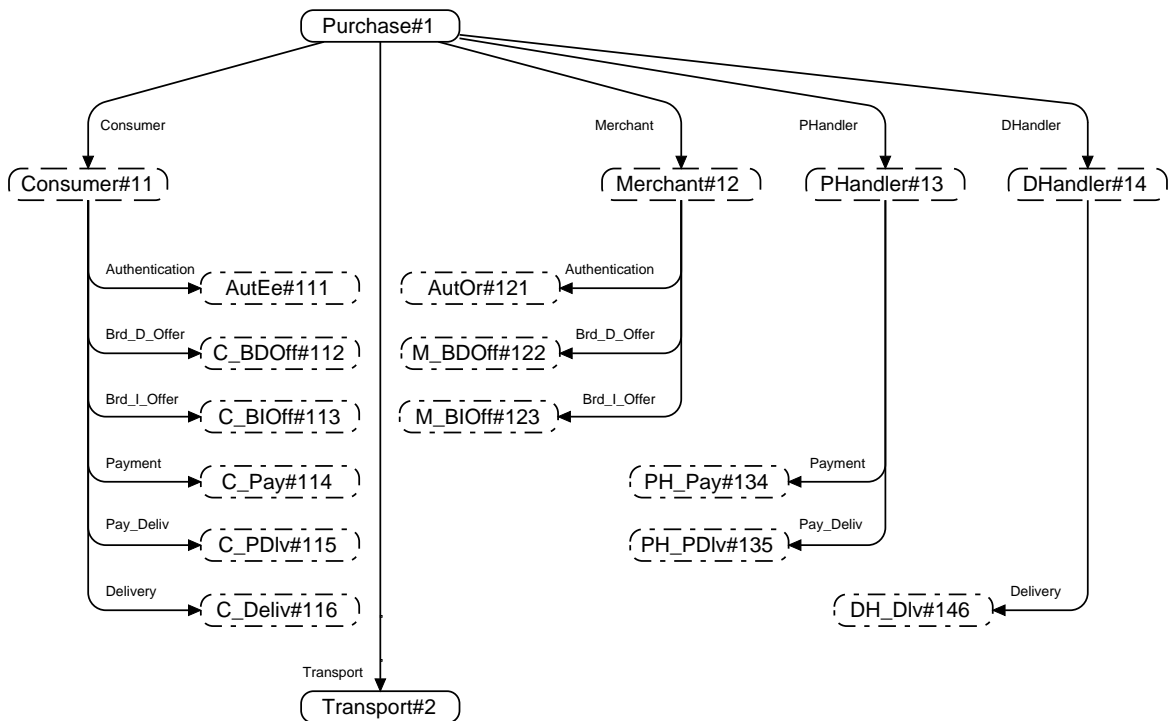


Fig. 2: The hierarchy page of the CPN Purchase transaction model.

The 12 subpages of the four trading role pages specifies the IOTP exchanges in a Purchase transaction. Baseline IOTP defines six document exchanges: *Authentication*, *Brand Dependent Offer*, *Brand Independent Offer*, *Payment*, *Delivery*, and *Payment with Delivery*. The Authentication exchange (subpages AutEe and AutOr) allows one trading role (the

authenticator) to authenticate another trading role (the authenticatee), i.e. to validate that the organisation is the one it claims to be. The Brand Dependent Offer exchange (subpages C_BDOff and M_BDOff) allows the Merchant to provide a list of payment brands to the Consumer for selection. The selected payment brand is to be used to carry out the payment. The Brand Independent Offer exchange (subpages C_BIO and M_BIO) allows the Merchant to specify (without Consumer selection) which payment brand is to be used for the payment. The Payment exchange (subpages C_Pay and PH_Pay) supports payment using some payment brand to be made by the consumer to the payment handler. The Payment with Delivery exchange (pages C_PDlv and PH_PDlv) supports a combined payment and delivery. The Delivery exchange (subpages C_Deliv and DH_Deliv) supports the delivery of goods. The IOTP exchanges are combined to implement the different transactions. For example, the Purchase transaction shown in Fig. 1 consists of a Brand Dependent Offer, a Payment, and a Delivery exchange. Considering that each IOTP exchange is defined as a bilateral business interaction between two trading roles, each exchange has been modelled as a pair of subpages – one for each trading role side. For example, the subpages C_Pay and PH_Pay represent the payment exchange carried out between the Consumer and the Payment Handler trading role. We will go into more detail with the specification of the IOTP exchanges in Sect. 5.

3.1 Trading Roles and Messages

An IOTP transaction consists of a set of *IOTP Messages* exchanged between trading roles. Figure 3 depicts the prime page Purchase. This page has a substitution transition for each of the four trading roles in a Purchase transaction. The substitution transition Transport represents the transport medium by means of which the trading roles communicate. Each trading role has two associated places modelling an input and an output message buffer.

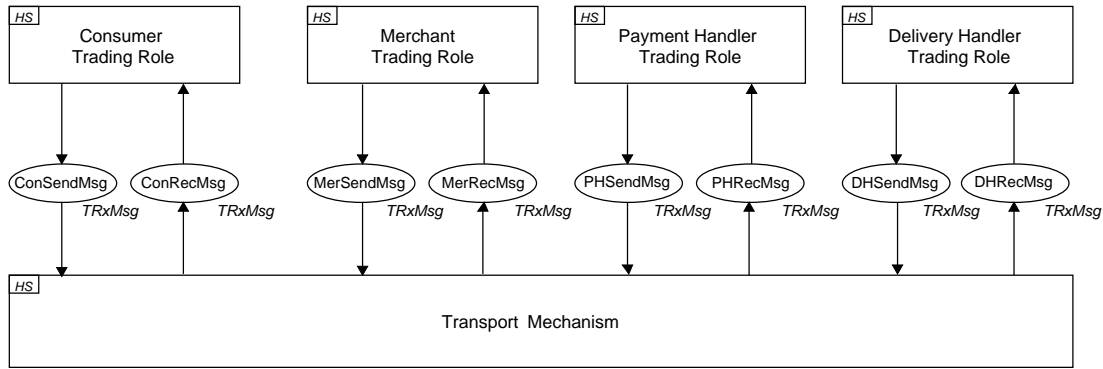


Fig. 3: Top level structure of IOTP Purchase transaction.

Figure 4 lists the definition of the colour sets used to model IOTP messages. The colour set TRxMsg (line 5) is used to model the message buffers containing IOTP messages to be transmitted or received. For an example, a token residing in a place that models a buffer for sending messages (e.g., ConSendMsg) specifies the message receiver trading role and the message itself. In the other case, a token residing in a place that models a buffer for receiving messages (e.g., ConRecvMsg) specifies the message sender trading role and the message itself.

IOTP messages are modelled by the colour set lotpMsg (line 4) which is derived from the XML (eXtensible Markup Language) definition of IOTP messages given in [3]. An IOTP

message consists of a list of so-called *Trading Blocks* modelled by the colour set `TradingBlk` (line 3). The colour set `ProcessState` (line 2) represents the *ProcessState* attribute of an *Authentication Status Block*. It contains two values, indicating whether the result of the Authentication exchange is successful (`CompletedOk`) or has failed (`Failed`). The representation of IOTP messages in the CPN model is a simplified and abstract representation of IOTP messages as specified in [3]. The reason is that not all attributes of an IOTP message are required at the level of abstraction where the trading transactions are being modelled here.

```

1  color TradingRole = with Consumer | Merchant | PHandler | DHandler;

2  color ProcessState = with CompletedOk | Failed;

3  color TradingBlk = union AuthReq + AuthResp + AuthStatus: ProcessState
                        + TPO + TPOSelection + OfferResp
                        + PayReq + PayExch + PayResp
                        + DeliveryReq + DeliveryResp
                        + Cancel;

4  color IotpMsg = list TradingBlk;

5  color TRxMsg = product TradingRole * IotpMsg;

```

Fig. 4: Colour sets for modelling IOTP trading roles and messages.

4 IOTP Transaction Layer

The IOTP transactions are implemented via combinations of IOTP exchanges at each trading role side. The Purchase transaction is the most complex trading transaction involving four trading roles and covers six IOTP exchanges. The transaction level pages in the CPN model are composed of four trading role pages. Each trading role page describes how the trading role operates in combining the IOTP exchanges to the Purchase transaction. In the following subsections we consider each of the trading roles at the transaction layer in detail.

4.1 Consumer Trading Role

The IOTP transactions are consumer oriented. Therefore, the Consumer is the most important trading role in the Purchase transaction. As defined in IOTP, the bilateral communications during the Purchase are always carried out between the Consumer and one of the other three trading roles. As a result, all the procedures for the Purchase transaction are performed at the Consumer side. Figure 5 depicts the page Consumer modelling the Consumer side of a Purchase transaction. IOTP defines the valid combinations of IOTP exchanges to implement an IOTP transaction. In Fig. 5, the IOTP exchanges that are part of a Purchase transaction are all represented by means of substitution transitions. A purchase transaction consists of an optional Authentication exchange, followed by an Offer exchange, and then either a Payment exchange followed by a Delivery exchange, a Payment exchange only, or a Payment with Delivery exchange.

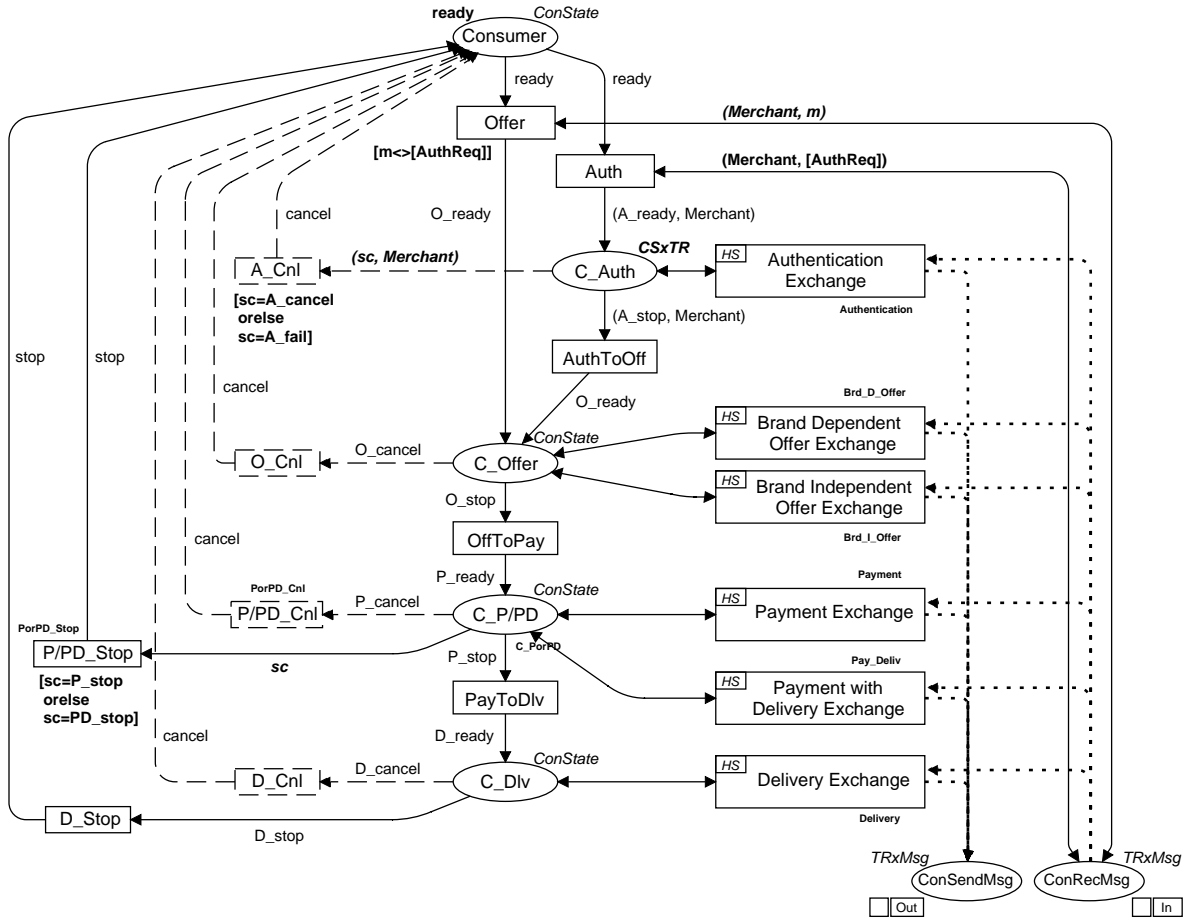


Fig. 5: Purchase transaction – Consumer trading role.

The five places *Consumer*, *C_Auth*, *C_Offer*, *C_PayDiv*, and *C_Deliv* are used to model the state of the Consumer during the execution of the Purchase transaction. The place *Consumer* (top) models the initial state *ready* and the two terminal states *cancel* and *stop* for a transaction. The other four places model the states of the Consumer in the six different IOTP exchanges involved in the transaction. Except for the arcs between the places and the substitution transitions, each input or output arc of these places has a constant in the arc expression. The constant represents the initial or the terminal state(s) of an IOTP exchange. The colour set *ConState* attached to these five places contains all the possible states of the Consumer, while the colour set *CSxTR* of place *C_Auth* also specifies the trading role corresponding to the other party. We will return to the definitions of these two colour sets when we explain how the IOTP exchanges have been modelled.

The Consumer is initially in the state *ready*. If the first IOTP message received from the Merchant is an Authentication Request message represented by a single-element list *[AuthReq]*, an Authentication exchange will start by putting a token with colour *(A_ready, Merchant)* in place *C_Auth*, indicating that the Consumer is now ready to be authenticated by the Merchant. Otherwise, the Consumer will directly start an Offer exchange by putting an *O_ready* token in place *C_Offer*. The occurrence of the transition *Auth* or *Offer* indicates that the Purchase transaction is carried out with an optional Authentication exchange. Moving

4.3 Payment Handler Trading Role

Figure 7 depicts page PHandler which specifies how the Payment Handler operates in a Purchase transaction. The Payment Handler is initially ready and will start either a Payment Exchange or a Payment with Delivery Exchange once receiving the message [PayReq] from the Consumer.

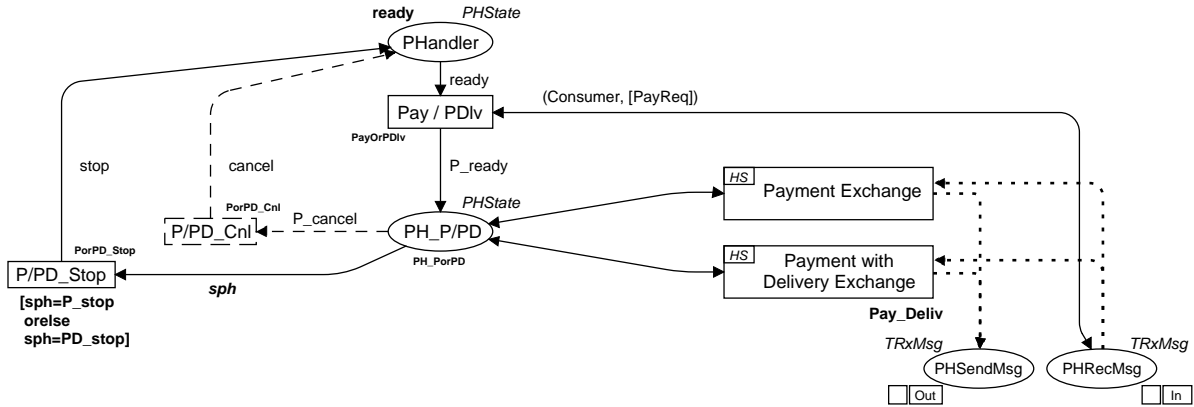


Fig. 7: Purchase transaction – Payment Handler trading role.

4.4 Delivery Handler Trading Role

Figure 8 depicts page DHandler which specifies how the Delivery Handler operates in a Purchase transaction. The Delivery Handler is initially ready and will start the Delivery Exchange as soon as the message [DeliveryReq] is sent from the Consumer indicating a delivery request.

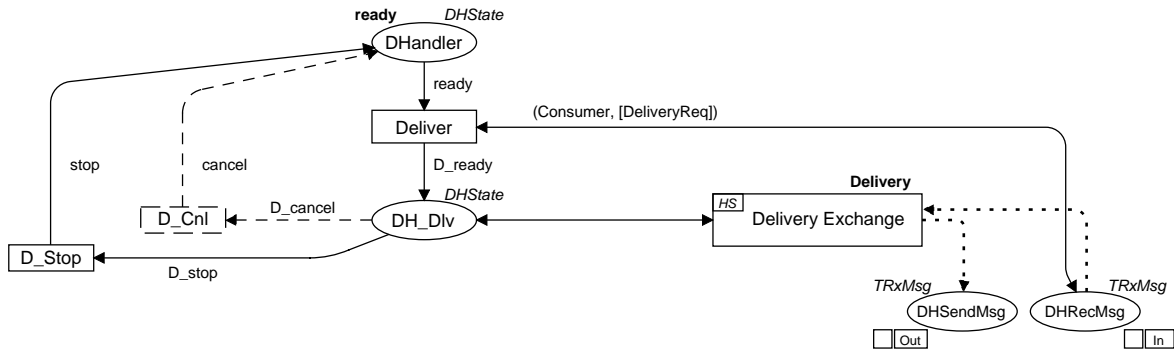


Fig. 8: Purchase transaction – Delivery Handler trading role.

5 IOTP Exchange Layer

We now consider the specification of IOTP exchanges. The same modelling approach has been applied to the six exchanges constituting a Purchase transaction. We therefore only give a detailed account and description of how the Payment exchange has been modelled.

5.1 State Definitions for Trading Roles

First we consider the declarations related to the states of the trading roles. It should be mentioned that there are no direct definitions of states for trading roles in the RFC [3], and they are thus indirectly derived from the RFC based on the messages sent and received by the trading roles. Figure 9 lists definitions of the colour sets relating to the states of trading roles. Colour set `State` (line 1) contains the states of all trading roles in a Purchase transaction. It has four subsets: `ConState` (line 2), `MerState` (line 3), `PHState` (line 4), and `DHState` (line 5). Each of them indicates the possible states of the corresponding trading role, i.e., Consumer, Merchant, Payment Handler, and Delivery Handler. The colour sets `CSxTR` and `MSxTR` contain a trading role component, and are used only for the Authentication exchange. This is needed because only one party (the Consumer) is fixed in an Authentication exchange whereas the other party could be any of the other trading roles. As part of its state the Consumer therefore needs to know who the other party is. In the following we explain each value in the colour set `State` in detail.

- `ready`, `cancel` or `stop` represents the initial state, the aborted terminal state, and the successful terminal state, respectively, of a trading role in a transaction. By adding a prefix to the above three states, corresponding states are obtained at the level of IOTP exchanges. For example, if the prefix is set to `A` for Authentication exchange, `A_ready`, `A_cancel` and `A_stop` then represent the initial state, and the two terminal states of a trading role in an Authentication exchange. Similarly, the prefix `O` stands for the two Offer exchanges, `P` stands for both the Payment and the Payment with Delivery exchanges, and `D` stands for the Delivery exchange. The only exception is the definition of the successful terminal state for the Payment with Delivery exchange. It is defined as `PD_stop`, which is different from `P_stop` for the Payment exchange.
- `A_fail` represents the unsuccessful terminal state for an Authentication exchange.
- `wait` represents the generic state indicating that a trading role is waiting for an IOTP message from another trading role.
- `proAuthReq` indicates that the trading role is in the state of processing an *Authentication Request Message*. Here, `pro` is a prefix standing for message processing, while `AuthReq` stands for the corresponding IOTP message. In this way, the various message processing states have been defined.

5.2 IOTP Payment Exchange

The Payment exchange allows a payment using some payment brand to be made by the Consumer to the Payment Handler. The RFC [3] defines a set of message processing guidelines for each exchange. As shown in Fig. 1, the messages exchanged in a successful Payment exchange consist of:

- A *Payment Request Message* sent from the Consumer to the Payment Handler to start the payment.
- A set of *Payment Exchange Messages* exchanged between the Consumer and the Payment Handler to process the payment.
- A *Payment Response Message* sent to the Consumer from the Payment Handler to complete the payment.

```

1  color State = with ready | wait | cancel | stop |
    A_ready | proAuthReq | proAuthRsp | A_cancel | A_stop | A_fail |
    O_ready | proTPO | proTPOSel | O_cancel | O_stop |
    P_ready | proPayReq | proPayExch | P_cancel | P_stop | PD_stop |
    D_ready | proDelivReq | D_cancel | D_stop;

2  color ConState = subset State with [ready, cancel, wait, stop,
    A_ready, proAuthReq, A_cancel, A_stop, A_fail,
    O_ready, proTPO, O_cancel, O_stop,
    P_ready, proPayExch, P_cancel, P_stop, PD_stop,
    D_ready, D_cancel, D_stop];

3  color MerState = subset State with [ready, cancel, wait, stop,
    A_ready, proAuthRsp, A_cancel, A_stop, A_fail,
    O_ready, proTPOSel, O_cancel, O_stop];

4  color PHState = subset State with [ready, cancel, wait, stop,
    P_ready, proPayReq, proPayExch, P_cancel, P_stop,
    PD_stop];

5  color DHState = subset State with [ready, cancel, wait, stop,
    D_ready, proDelivReq, D_cancel, D_stop];

6  color CSxTR = product ConState * TradingRole;
7  color MSxTR = product MerState * TradingRole;
8  var sc: ConState;          var sm: MerState;          var sph: PHState;

```

Fig. 9: Colour sets and variables for modelling states of trading roles.

During a Payment exchange, both the Consumer and the Payment Handler may cancel the payment by sending a *Cancel Message* to the other side. As a result, the payment should be cancelled at both sides resulting in a cancelled Payment exchange. The page specifying the Payment exchange has been developed according to the above message processing guidelines. It captures not only the successful Payment exchange, but all the possible executions of it.

Figure 10 depicts the page C_Pay (of Fig. 2) specifying the Consumer side of a Payment exchange. Figure 11 depicts the page PH_Pay specifying the Payment Handler side of a Payment exchange. In these two pages, each transition represents the event of sending or receiving a message. Then, the message to be transmitted contains not only the message content, but also the receiver trading role, while the message received contains the content together with the sender trading role.

We now illustrate how the pages shown in Fig. 10 and Fig. 11 reflect the Payment exchange. Both trading roles are initially in state P_ready. The payment starts when the Consumer sends [PayReq] to PHandler. On receiving [PayReq] from the Consumer, the Payment Handler checks the message and thus is in the state of proPayReq. If the result is valid, [PayExch] is sent to the Consumer. Otherwise, the Payment Handler will send a [Cancel] to cancel the payment. In Fig. 11 this is indicated by transition SendPayExch and SendPayCnl. In Fig. 10, identically named transitions are also used to indicate the same events are carried out by the Consumer

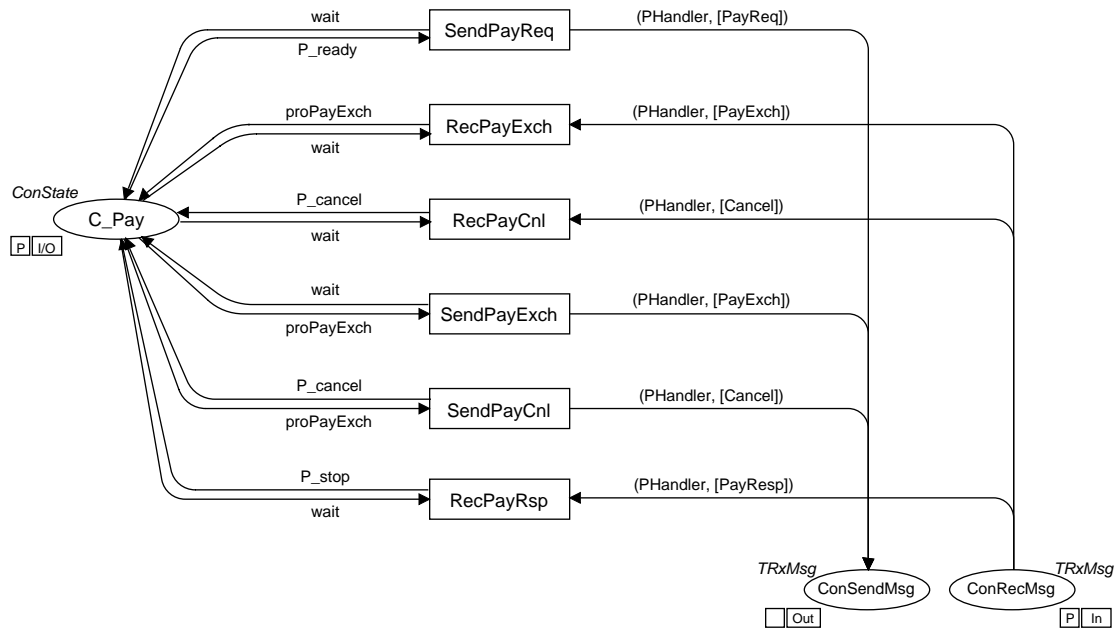


Fig. 10: The Payment Exchange – Consumer.

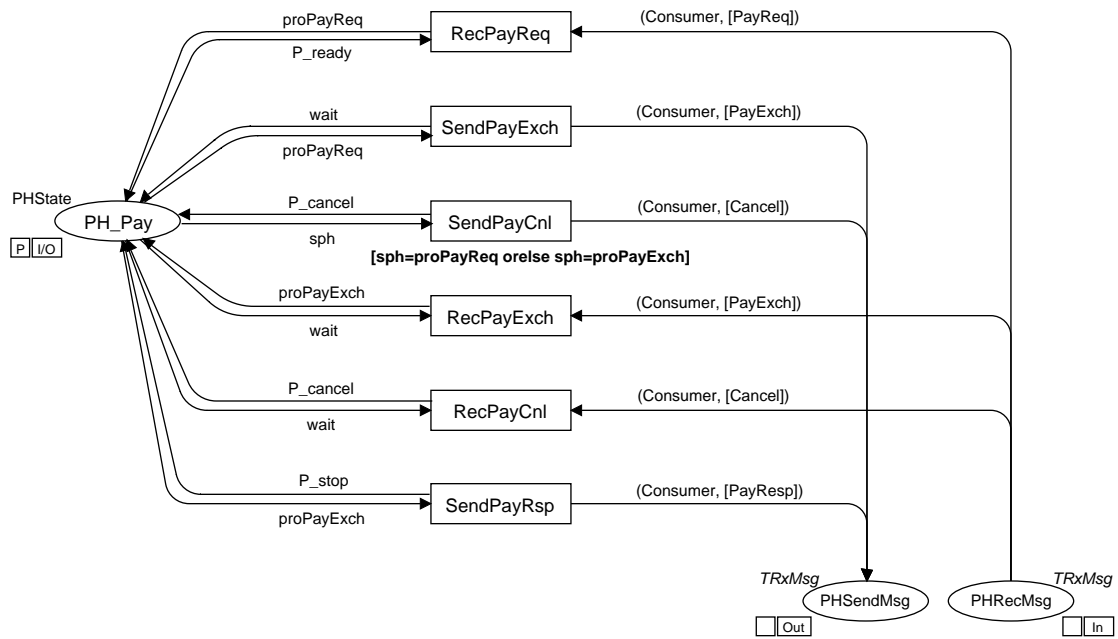


Fig. 11: The Payment Exchange – Payment Handler.

after receiving [PayExch], while the occurrence of RecPayCnl results in the cancellation of payment on the Consumer side after [cancel] is received. Then, on receiving the [PayExch], the Payment Handler may either send [PayResp] to complete the payment, or send [PayCnl] to cancel the payment. A successful Payment exchange will be completed on both sides when the Consumer receives [PayResp].

6 Validation of CPN IOTP Model

In this section, we apply simulation and state space analysis to validate the CPN model of the Purchase transaction. The simulation results are presented using *Message Sequence Charts* (MSCs) [1] as implemented in the MSC library of Design/CPN [12]. State space analysis has been conducted using the state space report produced by Design/CPN, and standard query functions. Simulation and state space analysis similar to the one presented for the Purchase transaction in this section has been conducted for the CPN models of the other IOTP transactions.

6.1 Simulation Analysis

The aim of the simulation analysis is to validate that the constructed CPN model of the Purchase transaction conforms with the specification of the Purchase transaction in the RFC. We do so by demonstrating that the CPN model of the Purchase transaction can generate the same set of message exchanges between trading roles as specified for the Purchase transaction in the RFC. The RFC specifies the message exchanges between trading roles in a Purchase transaction using MSCs. To make it easy to compare the message exchanges between trading roles as specified by the CPN model and as specified by the RFC, the CPN model has been instrumented so that MSCs displaying the message exchanges between trading roles can be automatically produced during a simulation. Figures 12-14 show three representative MSCs resulting from simulation of the CPN model. Each of them shows the message exchanges between the trading role entities (i.e., Consumer, Merchant, Payment Handler, and Delivery Handler) during a Purchase transaction. Time increases from the top of the chart to the bottom.

Figure 12 shows a successful completed Purchase transaction. The first three events are concerned with an Authentication exchange (labelled as event 1-3 in the figure). After that, a Brand Dependent Offer exchange is represented by the following 3 events (event 4-6). Next, a Payment exchange is carried out (event 7-10). Finally, the last two events shows the occurrence of a Delivery exchange (event 11-12). Therefore, Fig. 12 also provides the MSCs for these four IOTP exchanges, and each of them matches the corresponding MSCs from the RFC as shown in Fig. 1. However, Fig. 12 differs to Fig. 1 in two points. First, Fig. 12 contains the MSC for an initial Authentication exchange between Consumer and Merchant as part of a Purchase transaction, while there is no Authentication considered in Fig. 1. Second, messages outside the scope of IOTP are shown in Fig. 1 but become invisible in Fig. 12.

The specification of IOTP transactions in the RFC contains only the MSCs for those IOTP exchanges that are completed successfully without any cancellation during their execution. In contrast, the CPN model of the Purchase transaction specifies all the possible executions of a Purchase transaction. Figure 13 shows examples of a cancelled and a failed authentication exchange, and Figure 14 shows that the Payment exchange can be cancelled at two different stages during the transactions. These unsuccessful exchanges then result in the cancellation of the transaction they are part of.

6.2 State Space Analysis

The aim of the state space analysis is to investigate the behaviour of the CPN model as well as to validate and verify the functional correctness of the Purchase transaction. A full

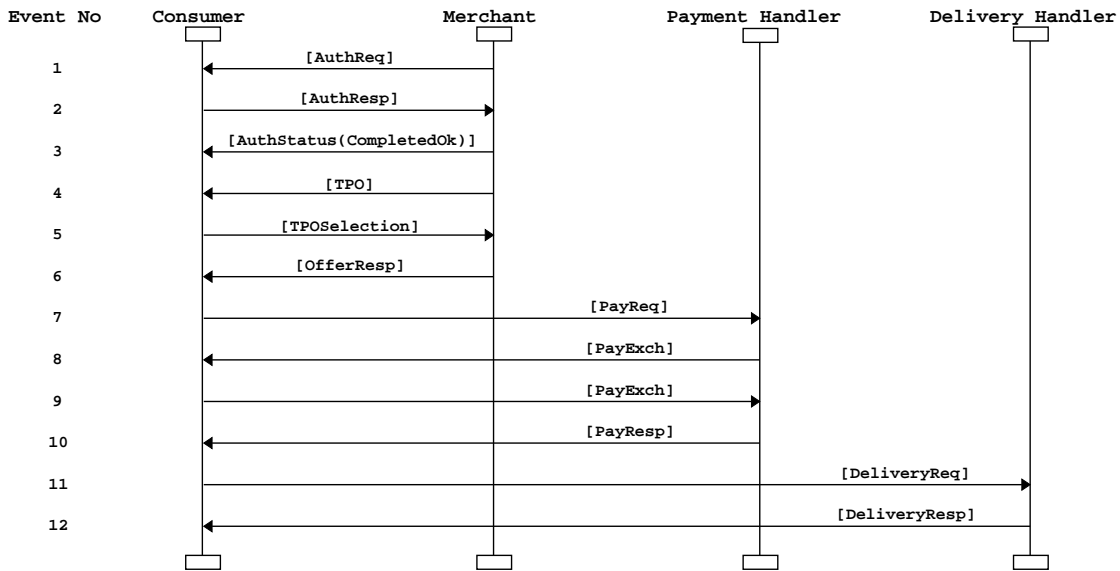


Fig. 12: The MSC for a successful Purchase transaction.

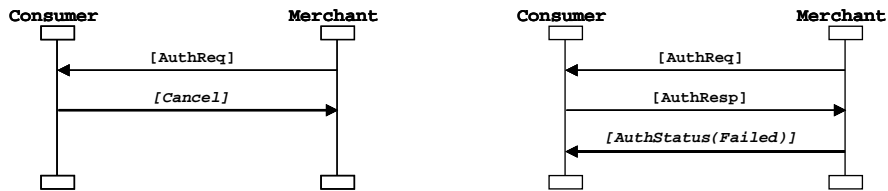


Fig. 13: The MSC for cancelled Authentication exchanges.

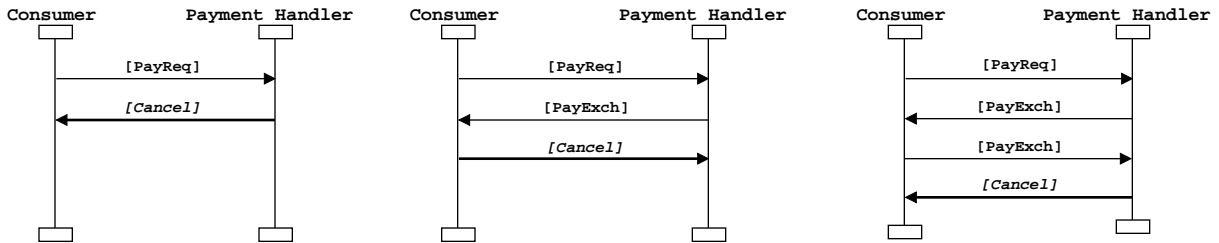


Fig. 14: The MSC for cancelled Payment exchanges.

state space report for the Purchase transaction model has been generated. The statistical information shows that the state space (OCC) has 244 nodes and 490 arcs.

Table 1 shows the home and the liveness properties and has been extracted from the state space report. There are five dead markings representing the terminal states of a Purchase transaction. Table 2 lists the corresponding states of each trading role in the five dead markings. Marking 164 and 243 represent the two possible successful executions of a Purchase transaction. In marking 164, the Purchase transaction is completed at the end of a Payment or a Payment with Delivery exchange between the Consumer and the Payment Handler. The

Delivery Handler is therefore never active and is in state ready at the end of the transaction. In marking 243, the purchase is completed with a Delivery exchange after the payment. The cancelled Purchase transactions are represented by the other three dead markings. Marking 57 corresponds to the state in which the transaction is cancelled during the Authentication exchange or one of the two Offer exchanges between the Consumer and the Merchant. The transaction can also be cancelled during the Payment exchange or the Payment with Delivery exchange, as indicated by marking 126. Finally, marking 244 represents the state where the transaction is cancelled during the Delivery exchange. Carefull inspection of Table 2 also shows that the states which the trading roles are in upon termination of the transaction are consistent. Therefore, investigating all the dead markings shows that the Purchase transaction terminates properly. The set of dead markings also constitutes a *home space*, i.e. it is always possible during the execution of the Purchase transaction to reach one of the dead markings. This has been checked using the query functions HomeSpace and ListDeadMarkings of the state space tool.

Table 1: Home and liveness properties.

Home Markings	None
Dead Markings	[57,126,164,243,244]
Dead Transitions Instances	None
Live Transitions Instances	None

Table 2: Trading role states in the dead markings.

Trading role	Dead marking				
	[57]	[126]	[164]	[243]	[244]
Consumer	1'cancel	1'cancel	1'stop	1'stop	1'cancel
Merchant	1'cancel	1'stop	1'stop	1'stop	1'stop
Payment Handler	1'ready	1'cancel	1'stop	1'stop	1'stop
Delivery Handler	1'ready	1'ready	1'ready	1'stop	1'cancel

Table 3 lists upper and lower integer bounds of the places modelling the message buffers (see Fig. 3). The boundedness results show that the minimal number of messages in each of the buffers is zero, which corresponds to the terminal states of the Purchase transaction. The maximum number of messages is 1 except that both the Consumer receiving buffer (ConRecMsg) and the Merchant sending buffer (MerSendMsg) may contain 2 at a time. The corresponding markings can be identified using the PredAllNodes query function of the state space tool, and an execution of the transaction leading to such marking can be found using the NodesInPath function. To understand why these two buffers may contain two messages at the same time consider the MSC in Fig. 12. The events 3 and 4 show that the Merchant sends the TPO message ([TPO]) directly after the successful Authentication status message ([AuthStatus(CompletedOk)]) with no confirmation message from the Consumer in between. The Consumer may receive the second message before processing the first one. Inspection of the multi-set bounds in the state space report shows that the trading roles do enter their expected states during the transaction, and that the messages exchanged between them are as expected.

Table 3: Upper and lower integer bounds.

Place	Upper	Lower	Place	Upper	Lower
ConRecMsg	2	0	MerRecMsg	1	0
ConSendMsg	1	0	MerSendMsg	2	0
DHRecMsg	1	0	PHRecMsg	1	0
DHSendMsg	1	0	PHSendMsg	1	0

7 IOTP Protocol Architecture

The RFC for IOTP does not contain a precise and detailed specification of the protocol architecture and the internal organisation of the individual layers. In this section we present our initial proposal for a protocol architecture of IOTP. The proposed protocol architecture is derived from the constructed CPN models of the trading transactions.

IOTP is clearly an application protocol in the context of the OSI reference model [2] possibly operating across different transport protocol services for transmission of IOTP messages. This is also evident from the prime page of the CPN model shown in Fig. 3. From the hierarchy page of the CPN model shown in Fig. 2, IOTP itself can be seen as consisting of two layers: a *transaction layer* and a *exchange layer*, where the transaction layer uses the services provided by the exchange layer to implement the trading transactions.

IOTP protocol entities are not designed to work solely as stand alone applications. An IOTP protocol entity will typically be embedded in other applications such as HTTP clients and HTTP servers when, e.g., the Consumer trading role entity is implemented in a HTTP web browser and the Merchant trading role entity is implemented in a HTTP web server for trading across the Internet. Moreover, IOTP will in most cases operate across the same transport service as the application it is part of. This transport service will typically vary depending on the application and the underlying communication medium. This suggests the existence of a *transport adaption layer*. This layer makes it possible to adapt the transport service of the specific application to the service required by IOTP.

Figure 15 shows the proposed protocol architecture of IOTP with the four main protocol layers as identified above. In the following we discuss the service provided by individual layers as well as their internal organisation in more detail.

Transactions Layer. This layer implements the IOTP trading and infrastructure transactions. For baseline IOTP this layer implements *Deposit*, *Withdrawal*, *Purchase*, *Refund*, *Value Exchange*, *Inquiry*, and *Ping* transactions.

Exchange Layer. This layer implements the IOTP exchanges. For baseline IOTP this layer implements *Authentication*, *Brand Dependent Offer*, *Brand Independent Offer*, *Payment*, *Delivery*, and *Payment with Delivery* exchanges. The CPN model presented in this paper specifies in detail how the individual trading transactions can be implemented as combinations of these exchanges.

Payment Sublayer. IOTP can support different payment instruments (such as VISA, MasterCard, and Mondex Card) by encapsulating the underlying payment protocols (such as

SET, Mondex VTP) during the payment-related exchanges. The payment sublayer has been defined as a sublayer embedded in the exchange layer since the *payment scheme component* defined in IOTP will be used to encapsulate the payment protocol data, e.g., a SET message, during a Payment exchange. One component of the payment sublayer is the so-called IOTP *payment bridges* specified in the Internet Draft for IOTP Payment API [6]. These payment bridges specify the interface and interaction between IOTP exchanges and the payment system.

Application Transport Adaption Layer. This layer interfaces IOTP to the underlying transport layer and transport service. This layer will ensure that the IOTP messages, which are well-formed XML documents, are carried successfully between the trading role entities. Typically, the application transport adaptation layer includes the mechanisms which support the mapping of the IOTP data format to the underlying transport layer, e.g., an XML to HTTP mapping.

Application Transport Layer. This is the basic transport service on top of which the IOTP entities are operating. This layer could be TCP/IP as well as other higher level transport protocols such as HTTP and SMTP. It might also be the Wireless Application Protocol (WAP) [5] transport service if IOTP is used in wireless e-commerce such as mobile commerce (m-commerce) applications. This layer can also provide session layer services such as those provided by Secure Socket Layer (SSL) for encryption.

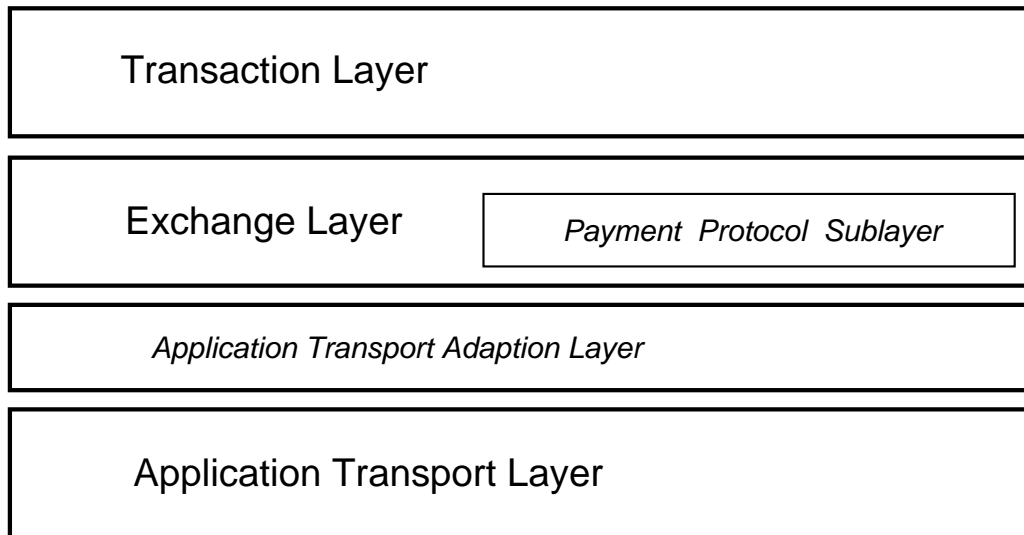


Fig. 15: The layered IOTP architecture.

The protocol architecture above has only one concept of *IOTP exchanges* – constituting the exchange layer. This contrast with the RFC which operators with two notions of exchanges: *document Exchanges* and *trading exchanges*. The CPN models of the IOTP transactions and the validation presented in Sect. 6 show that the concept of *document exchanges* suffices, since it can implement all the IOTP trading transactions. As a simplification of the IOTP specification we therefore propose to eliminate the concept of *trading exchanges* since it is not required from a protocol perspective.

8 Conclusions

We have presented a hierarchical CPN model of IOTP based on RFC 2801. The validation results demonstrate that our CPN IOTP model conforms to the specification given in the RFC. We have also proposed a complete and simplified architecture for IOTP based on the constructed CPN models. In addition, we propose the merging of the two unclarified notions of *trading exchanges* and *document exchanges* into one concept which we call *IOTP exchanges* leading to a simpler specification and architecture for IOTP.

For the current modelling of the IOTP transactions, we have a number of simplifying assumptions and abstractions. These simplifications are primarily related to IOTP error handling. For example, we have only investigated IOTP over a perfect transport medium without any message loss. Besides the error handling, the arbitrary cancellation during a transaction has not yet been taken into consideration. As part of future work we plan to include more internal operations in each layer such as the error handling and the arbitrary cancellation in our CPN model. However, the assumptions and simplifications will not affect the proposed protocol architecture.

Investigating the IOTP service specification will be another challenging part of future work, since there are no concepts of IOTP service defined in the RFC. The IOTP architecture proposed in this paper not only leads to a protocol specification, but also presents a first step towards developing a service specification for IOTP.

References

1. ITU Recommendation Z.120, *Message Sequence Chart*, 1992.
2. ITU-T Recommendation X.200, *Information Technology - Open Systems Interaction - Basic Reference Model*, July 1994.
3. D. Burdett. *Internet Open Trading Protocol - IOTP. RFC 2801*. IETF Trade Working Group, April 2000. Version 1.0.
4. D. Burdett, D. Eastlake, and M. Goncalves. *Internet Open Trading Protocol*. McGraw-Hill, 2000.
5. WAP Forum. *Wireless Application Protocol Architecture Specification*. Available via <http://www.wapforum.org/>, 30 April 1998.
6. W. Hans, Y. Kawatsura, and M. Hiroya. *Payment API for v1.0 Internet Open Trading Protocol (IOTP)*. IETF Trade Working Group, April 2001.
7. The Internet Engineering Task Force - IETF. <http://www.ietf.org/>.
8. Hitachi Research Topics IOTP. http://www.hitachi.co.jp/english/topics/t_iotp/iotp.html.
9. JOTP Open Trading Protocol Toolkit For Java. <http://www.livebiz.com/>.
10. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume-1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, 1992.
11. L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98-132, 1998. Springer-Verlag.
12. Design/CPN Message Sequence Chart Library. Department of Computer Science, University of Aarhus, Denmark., 1998. Version 1.1. Available via <http://www.daimi.au.dk/designCPN/>.
13. Design/CPN online. Available via <http://www.daimi.au.dk/designCPN/>.
14. Hitachi Smiles. http://www.hitachi.co.jp/Div/nfs/whats_new/smiles.html.
15. Mondex USA. <http://www.mondexusa.com/html/content/secur/security.htm>.
16. Visa and MasterCard. *SET Secure Electronic Transaction Specification. Volume-1,2,3*, May 1997. Version 1.0.
17. V. Zwass. Structure and Macro-Level Impacts of Electronic Commerce: From Technological Infrastructure to Electronic Marketplaces. In *Foundations of Information Systems*. McGraw-Hill, 1998.