Applications of Coloured Petri Nets for Functional Validation of Protocol Designs

Lars M. Kristensen[1] and Kent Inge Fagerland Simonsen[1,2]

[1] Department of Computer Engineering, Bergen University College, Norway
Email: {lmkr,kifs}@hib.no
[2] DTU Informatics, Technical University of Denmark, Denmark
Email: kisi@imm.dtu.dk

Abstract. Communication protocols constitute central building blocks in most modern IT systems as they define components, rules, and languages that make data communication possible. The development of correct protocols is a challenging engineering discipline, making modelling and validation of protocol design an important application domain for Coloured Petri Nets (CPNs). We illustrate the practical application of CPNs for protocol validation by focusing on selected aspects of four recent projects involving industrial-sized protocols. These projects demonstrate how CPNs can be used to model protocol elements and improve protocol specifications, how state space exploration can be used to verify protocol properties, and how behavioural visualisation in combination with a CPN model provides an effective way of rapidly constructing an executable prototype of a protocol design.

1 Introduction

Communication protocols play an important role in most IT systems. A prominent example is the vast amount of web applications that are in use today for, e.g., online banking, shopping, government administration, and entertainment. The services provided by these applications all rely on the protocols governing the operation of the Internet. Other examples are telecommunication systems, logistic systems with sensors and actuators, and control systems in vehicles. All these systems rely heavily on communication and synchronisation between concurrently executing software components and subsystems. As protocols are to support still more complex services that are critical to both the operation of companies and the everyday life of citizens, it is important that they are working correctly from the beginning of deployment.

Protocol engineering [57] typically involves a specification of the *service* that the protocol is to provide. Through a synthesis or design step, a protocol design is developed with the aim of providing the desired service. For protocol design, *functional* and *performance* validation can be conducted to investigate and reason about the properties of the design. Functional validation focuses on the logical correctness of the protocol such as the absence of deadlocks and livelocks, that a request is always followed by a response, or whether the proposed protocol design provides the desired services. Performance validation is concerned with properties such as delays, throughput, and response time. Eventually, the protocol design is implemented which may then be subject to further testing.

Protocol design is in many cases a challenging task. One reason for this is that the execution of a protocol can proceed in different ways, e.g., depending on which messages are lost in transmission, the scheduling of the protocol entities, the time at which events are received from the environment of the protocol, and the execution path taken by the protocol entities. Another reason is that a protocol by nature involves independently scheduled entities which makes testing and reproduction of executions difficult. All this means that protocols often have a very large number of possible executions. In this process, it is easy for a protocol engineer to overlook important interaction patterns which may in turn lead to gaps or malfunction of the protocol.

The specification of the protocol service and the protocol design is, in many cases, based on natural language descriptions. One example of this is the Request for Comments (RFC) documents published by the Internet Engineering Task Force (IETF) [33]. Natural language specifications of protocols often have many issues that needs to be resolved before a properly working implementation can be obtained. One source of issues is the fact that such specifications are inherently ambiguous making it difficult to achieve inter-operability between independent implementations. Another source of issues to resolve is that the specifications are often incomplete in that the behaviour of the protocol is not described for all cases.

The challenges outlined above has made protocols a prominent application domain for formal description techniques [32], including Petri Nets [69, 71]. In this paper we concentrate on the use of Coloured Petri Nets (CPNs) [35, 38, 39] for modelling and functional validation of protocol designs. The CPN modelling language belongs to the family of High-level Petri Nets and combine Petri Nets with the Standard ML (SML) programming language [74]. Petri Nets provide the foundation of the graphical notation and the semantical foundation for modelling concurrency, synchronisation, and communication. The functional programming language SML provides primitives for representing sequential aspects of protocols (such as data manipulation) and for creating compact and parameterisable models. Formal modelling and validation with CPNs are supported by CPN Tools [70] which provides support for construction, simulation, functional and simulation-based performance analysis of CPN models. In addition, animation libraries are available [85] which makes it possible to visualise the execution of protocols using, e.g., message sequence charts.

The advantage of CPNs (and formal description techniques in general) is that they are based on the construction of executable models that make it possible to observe and experiment with the protocol design prior to implementation and deployment using, e.g., simulation. This typically leads to more complete protocol specifications since the model will not be fully operational until all parts of the protocol have been (at least abstractly) specified. Furthermore, the construction of a formal and executable model helps identify and resolve ambiguities that may be present in a natural language specification. Another advantage is the support for model abstractions that makes it possible to specify the operation of the protocol without being concerned with implementation details such as message layout. A model also makes it possible to explore larger scenarios of a protocol system than what is in many cases practically possible in a laboratory.

Verification of behavioural properties of protocols with CPNs [44] is supported by explicit state space exploration [3]. In its simplest form this approach involves computing a directed graph where the nodes corresponds to the set of reachable states of the CPN model and the arcs represent occurrences of events causing state changes. The construction of the state space guarantees complete coverage of all executions and provides a highly systematic error-detection technique that make it possible to automatically (i.e., algorithmically) check whether a protocol has a formally stated desired property. In addition, state space methods have the advantage that counter examples (error-traces) can be automatically synthesised if the protocol does not satisfy a given property. The main disadvantage is the inherent state explosion problem [77] and in response a multitude of techniques have been devised in the context of CPNs and also more generally to alleviate the state explosion problem.

A concrete example illustrating the above advantages of formal modelling and verification is a project conducted at Ericsson Telebit A/S on the design of the edge router discovery protocol (ERDP) [45]. In this project, CPN modelling and state space exploration were applied as part of the protocol design project. Table 1 categorises and enumerates the issues encountered during two review phases (Review 1 and Review 2) of the protocol design. The issues were identified in the process of constructing the CPN model, performing single-step executions of the CPN model, and engaging in discussions of the CPN model with the protocol engineers at Ericsson. Subsequently, additional subtle behavioural errors causing incorrect synchronisation and livelocks were detected in the protocol design using basic state space exploration.

The project at Ericsson highlights the benefits of a formal modelling and validation approach. Furthermore, the project emphasised the benefits of the model

Table 1. ERDP project [45] – design issues identified in the modelling phase.

Category	Review 1	Review 2	Total
Errors in protocol specification/operation	2	7	9 issues
Incompleteness and ambiguity in specification	3	6	9 issues
Simplifications of protocol operation	2	0	2 issues
Additions to the protocol operation	4	0	4 issues
Total	11	13	24 issues

construction phase which is often underestimated (or not reported) in literature on protocol validation. As illustrated by the ERDP project, the modelling phase itself lead to significant insight into the protocol design, and contributed to a simpler and more complete protocol design.

The purpose of this paper is to provide an introduction to, and an overview of, how CPNs have been applied for practical validation of protocol designs. We approach this by presenting selected parts of CPN models and associated results extracted from projects conducted in an industrial context with industrial-sized protocols. More specifically, we present in the following sections the application of the CPN modelling language, tools, and techniques for functional validation of the following protocols:

DYMO Routing Protocol. Section 2 presents a CPN model constructed in a project [18] on modelling and validating the Dynamic On-Demand Routing Protocol for Mobile Ad-hoc Networks (DYMO) [9]. DYMO is a routing protocol for mobile ad-hoc networks being developed by the MANET working group of the IETF. The DYMO CPN model is used to introduce the basic constructs of the CPN modelling language and show how they can be used to model the elements of protocols.

Generic Access Networks Architecture. Section 3 shows how CPN modelling was used at TietoEnator A/S to specify the detailed usage of protocol software via specialisation of the Generic Access Network (GAN) architecture [1] developed by the 3rd Generation Partnership Project (3GPP) [21]. This section shows how basic state space exploration is typically conducted as a first step for the initial verification of a protocol design.

Routing Interoperability Protocol. Section 4 illustrates how CPN modelling were used at Ericsson Telebit A/S to specify the operation of a routing interoperability protocol (RIP) [51] for routing (packets) between core networks and mobile ad-hoc networks. This section shows how application-specific behavioural visualisation can be applied on top of CPN models to obtain a first executable prototype of the protocol design allowing for early experiments and for presentation to customers and management with the aim of soliciting protocol design requirements.

Edge Router Discovery Protocol. Section 5 explains how CPN modelling was used in the project [45] at Ericsson Telebit A/S on the design of the ERDP protocol. ERDP is an IPv6-based protocol allowing edge routers to configure gateways in mobile ad-hoc networks with IP address prefixes. The section shows how modelling, state space exploration, and behavioural visualisation all contributed to identify and resolve design issues and errors during development.

Section 6 contains conclusions, provides further references to projects where CPNs have been applied for protocol validation, and outlines directions for future work. The reader is assumed to be familiar with the basic ideas of Petri Nets [71] and explicit state space exploration. The reader is referred to [38] for a comprehensive introduction to CPNs and state space exploration of CPN models.

2 DYMO and Protocol Modelling with CPNs

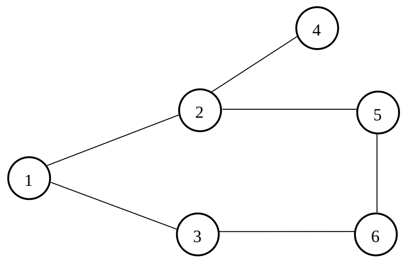
Modelling a protocol involves developing a representation of the *messages* (or packets) exchanged between the *protocol entities*, the *procedure rules* and *internal state* of the protocol entities guarding the processing of messages, and developing a model of the *environment* in which the protocol is being executed. The environment model typically encompass an abstract representation of the communication medium (or channel) over which the protocol operates. We illustrate how these protocol elements can be represented in the CPN modelling language using the Dynamic On-demand Routing Protocol (DYMO) [9] for mobile ad-hoc networks as an example. This protocol formed the basis of the modelling and validation project reported [18] in which CPN Tools was applied to model and improve an earlier version of the DYMO protocol. A CPN model of the DYMO protocol has also been developed in [8] where a considerable more compact CPN model of the DYMO protocol directly targeting state space exploration is developed.

2.1 The DYMO Protocol

A *mobile ad-hoc network* (MANET) comprises a collection of mobile nodes, such as laptops, personal digital assistants, and mobile phones, capable of establishing a communication infrastructure for their common use. Ad-hoc networking differs from conventional networks in that the nodes operate in a fully self-configuring, autonomous and distributed manner, without any preexisting communication infrastructure such as base stations and routers. Network layer and routing protocols for ad-hoc networking (including the DYMO protocol) are currently under development by the IETF MANET working group.

The operation of the DYMO protocol consists of two parts: *route discovery* and *route maintenance*. Route discovery is used to establish routes between nodes and begins with an *originator node* multi-casting a Route Request (RREQ) message to all nodes in its immediate range. A RREQ message has a *sequence number* to enable other nodes in the network to judge the freshness of the route request. The ad-hoc network is then flooded with RREQs until the request reaches the *target node* (provided that there exists a path from the originating node to the target node). The target node replies with a Route Reply (RREP) message unicasted hop-by-hop back to the originator node. The route discovery procedure is requested by the Internet Protocol (IP) layer on a node when it receives an IP packet for transmission and does not have a route in its routing table to the target node.

Figure 1(left) depicts the topology of a MANET consisting of six nodes numbered 1–6. An edge between two nodes indicates that the nodes are within direct transmission range. In this case, we assume that all communication links are symmetric. Figure 1(right) (to be discussed below) lists for each node the *routing table* entries created as a result of executing a routing discovery procedure with node 1 as the originator node and node 6 as the target node. The routing table entries in Fig. 1(right) are specified as a pair $(target, nexthop)$. The second



Node	RREQ	RREP
1		(6,3)
2	(1,1)	
3	(1,1)	(6,6)
4	(1,2)	
5	(1,2)	
6	(1,3)	

Fig. 1. Example MANET topology (left) and routing table entries (right).

column specifies the entries that are created as a result of a node receiving the RREQ. The third column lists the entries created as a result of receiving the corresponding RREP. When explaining the operation of the DYMO CPN model below, we will use the scenario in Fig. 1 as a running example.

The *message sequence chart* (MSC) in Fig. 2 depicts one possible exchange of messages in the DYMO protocol when the originating node 1 establishes a route to target node 6 in the topology in Fig 1(left). Solid arcs represent multi-cast transmission and dashed arcs represent unicast transmission. In the MSC, node 1 multi-casts a RREQ which is received by nodes 2 and 3. When receiving the RREQ from node 1, nodes 2 and 3 create an entry in their routing table specifying a route back to the originator node 1. Since nodes 2 and 3 are not the target of the RREQ they both multi-cast the received RREQ to their neighbours (nodes 1, 4 and 5, and nodes 1 and 6, respectively). Node 1 discards these messages as it was the originator of the RREQ. When nodes 4 and 5 receive the RREQ they add an entry to their routing table specifying that the originator node 1 can be reached via node 2. When node 6 receives the RREQ from node 3, it discovers that it is the target node of the RREQ, adds an entry to its routing table specifying that node 1 can be reached via node 3, and unicasts a RREP back to node 3. When node 3 receives the RREP it adds an entry to its routing table stating that node 6 is within direct range, and use its entry in the routing table that was created when the RREQ was received to unicast the RREP to node 1. Upon receiving the RREP from node 3, node 1 adds an entry to its routing table specifying that node 6 can be reached using node 3 as the next hop. The RREQ is also multi-casted by node 4, but when node 2 receives it again, it will be discarded by node 2 because it has already processed the RREQ message once. Node 5 also multi-casts the RREQ, but nodes 2 and 6 also discard the RREQ message as it has already been received once. From Fig. 1(right) it can be seen that upon completion of the route discovery procedure, a bidirectional route has been discovered and established between node 1 and node 6 using node 3 as an intermediate hop.

The topology of a MANET changes over time because of the mobility of the nodes. DYMO nodes therefore perform *route maintenance* where each node monitors the links to the nodes it is directly connected to. The DYMO protocol has a mechanism to notify nodes about routes that become broken due to nodes

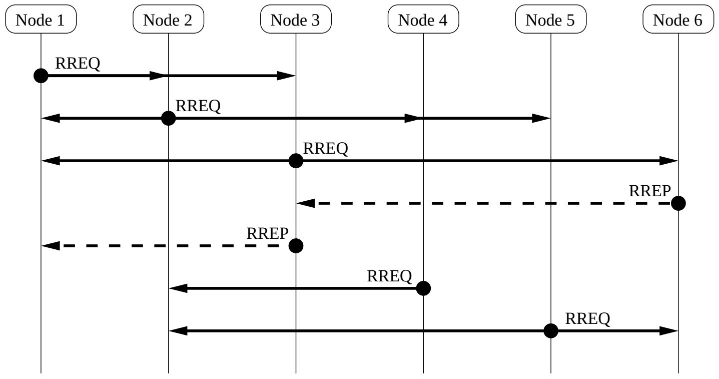


Fig. 2. Message exchange scenario showing DYMO route discovery procedure.

moving out of range of each other. This is done by sending Route Error (RERR) messages which have the effect of informing nodes using the broken route that a new route discovery is needed in order to reestablish a communication path.

2.2 DYMO Model Overview and Message Modelling

The DYMO CPN model is a hierarchical model organised in 14 *modules*. Figure 3 shows the *module hierarchy* of the CPN model. Each node in Fig. 3 corresponds to a *module* with System representing the top-level module of the CPN model. An arc leading from one module to another indicates that the latter is a *submodule* of the former. The model is organised into two main parts. The DYMOProtocol module and its nine submodules model the DYMO protocol entities including the internal state of the protocol entities and the procedure rules for receiving messages, internal processing, and sending of messages. The MobileWirelessNetwork module and its two submodules models the environment for the DYMO protocol. This includes the modelling of how messages are transmitted over a wireless link and the modelling of how the mobility of the nodes affect the current topology of the network. The division of the model into submodules reflects the structure of the DYMO specification [10] and hence maintain a close structural relationship between the natural language specification and the formal CPN model. The CPN model does not capture the transmission of payload from the application layer as the focus of the model is on the route establishment and maintenance of the DYMO protocol.

The top-level module System is shown in Fig. 4 and is used to connect the two main parts of the model. It corresponds to the System node in Fig. 3. The module has two *substitution transitions* drawn as rectangles with double-line borders. Each of the substitution transitions has an associated tag positioned next to them specifying the name of the associated submodule. The DYMO-Protocol substitution transition has the DYMOProtocol module as its associated submodule, and the MobileWirelessNetwork substitution transition has the mod-

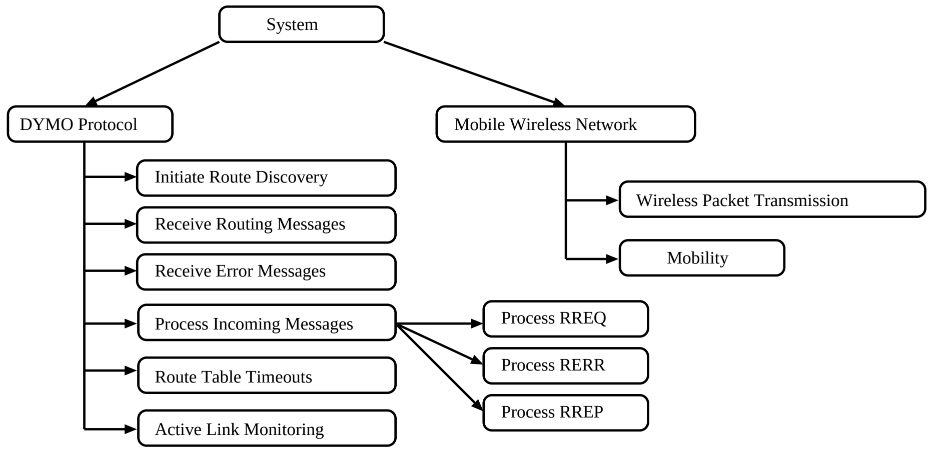


Fig. 3. Module hierarchy for the DYMO CPN model.

ule MobileWirelessNetwork as its associated submodule. In this model, the substitution transition has the same name as its associated submodule (but this is not generally required).

The two *socket places* DYMOToNetwork and NetworkToDYMO connected to the substitution transition DYMOProtocol are used to model the interaction between the DYMO protocol and the MANET environment as represented by the submodules of the MobileWirelessNetwork substitution transition. The socket place LinkState is used to model the *active link monitoring* that nodes perform to check which neighbour nodes are still reachable. When the DYMO protocol module sends a message, it will appear as a token representing a network packet on the socket place DYMOToNetwork. Similarly, a network packet to be received by the DYMO protocol module will appear as a token on the NetworkToDYMO socket place. Each of the socket places in Fig. 4 (places connected to a substitution transition) is associated with a *port place* in the submodule associated with the substitution transition that the socket place is connected to. The association between a socket and a port place has the effect that the port and the socket places will always have identical markings (tokens). An arc leading to a socket place from a substitution transition means that transitions on the submodule associated with the substitution transitions will add tokens on this place. Analogously, an arc leading from a socket place to a substitution transition means that transitions on the submodule will remove tokens from this place.

The colour set (data types) of each place determining the kind of tokens that can reside on the place is written below each place. The colour set declarations used in Fig. 4 is provided in Fig. 5. A record colour set is used for representing the packets transmitted over the wireless links. A `NetworkPacket` consists of a source (field `src`), a destination (field `dest`), and some data (payload). The DYMO messages are designed to be carried in User Datagram Protocol (UDP) datagrams. This means that the network packets are abstract representations

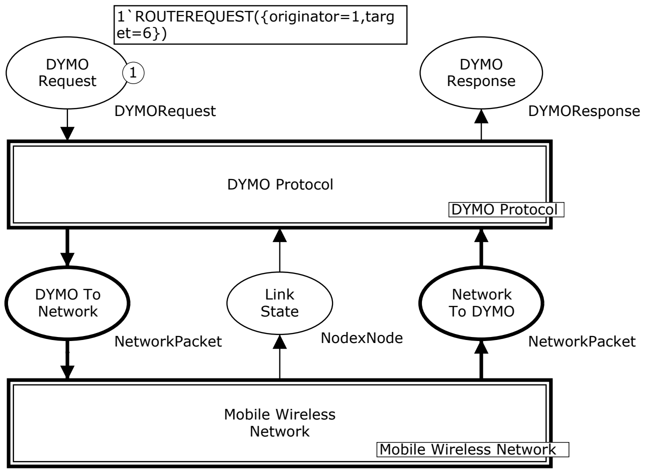


Fig. 4. Top-level System module of the DYMO CPN model.

of IP/UDP datagrams. The model abstracts from all fields in the IP and UDP datagrams (except source and destination fields) as only these impact the DYMO protocol logic. The source and destination of a network packet are modelled by the `IPAddr` colour set. There are two kinds of IP addresses in the model: `UNICAST` addresses and the `LL_MANET_ROUTERS` multi-cast address. The multi-cast address is used, e.g., in route discovery when a node is sending a RREQ to all its neighbouring nodes. Unicast addresses are used as source of network packets and, e.g., as destinations in RREP messages. A unicast address is represented as an integer from the colour set `Node`. Hence, the model abstracts from real IP addresses and identify nodes (communication interfaces) using integers in the interval $[1; N]$ where `N` is a model parameter specifying the number of nodes in the MANET.

The two places DYMORequest and DYMOResponse in Fig. 4 are used to interact with the service provided by the DYMO protocol. A route discovery for a specific destination is requested by putting a token on the DYMORequest place and a DYMO response to a route discovery request is then provided by DYMO as a token via the DYMOResponse place. The colour sets `DYMORequest` (see Fig. 5) specifies the identity of the `originator` node requesting the route and the identity of the `target` node to which a route is to be discovered. Similarly, a `DYMOResponse` message contains a specification of the `originator`, the `target`, and a boolean `status` specifying whether the route discovery was successful. The colour sets `DYMORequest` and `DYMOResponse` are defined as union types to make it easy to later extend the model with additional requests and responses. By setting the *initial marking* of the place DYMORequest, it can be controlled which route discovery requests are to be made.

The small circles and associated boxes in Fig. 4 show the *current marking* of the CPN model. The small circle positioned inside a place indicates the number of

```
(* --- Nodes and abstract IP/UDP messages --- *)
colset Node          = int with 0 .. N;
colset IPAddr        = union UNICAST : Node + LL_MANET_ROUTERS;

colset NetworkPacket = record src  : IPAddr * dest : IPAddr *
                              data : DYMOMessage;

(* --- DYMO service --- *)
colset RouteRequest = record originator : Node * target : Node;

colset DYMORequest  = union ROUTEREQUEST : RouteRequest;

colset RouteResponse = record originator : Node * target : Node *
                              status     : BOOL;

colset DYMOResponse  = union ROUTERESPONSE : RouteResponse;
```

Fig. 5. Colour set declarations for nodes, network packets, and DYMO service.

tokens on the place in the current marking. In Fig. 4, there is a single token on the place DYMORequest with colour `ROUTEREQUEST({originator=1,target=6})` as specified in the box positioned next to the small circle. This marking corresponds to the DYMO protocol being requested to establish a route from node 1 to node 6 as considered in the scenario in Fig. 1.

2.3 The DYMO Protocol Entities

The top-level module for the DYMO protocol part of the CPN model is the DYMOProtocol module shown in Fig. 6. The module has five substitution transitions modelling initiating route requests (substitution transition InitiateRouteDiscovery), reception of RREQ and RREP messages (substitution transition ReceiveRoutingMessages), the reception of RERRs (substitution transition ReceiveErrorMessages), processing of incoming messages (substitution transition ProcessIncomingMessages), and timer management associated with the routing table entries (substitution transition RouteTableEntryTimeouts). The places DYMORequest, LinkState, and NetworkToDYMO are *input port places* of the module as indicated by the In tag positioned next to them. Each of these places are associated with the accordingly named socket places in Fig. 4. Similarly, the places DYMOToNetwork and DYMOResponse are *output port places* as indicated by the Out tag positioned next to them, and they are associated to the accordingly named socket places in Fig. 4

All submodules of the substitution transitions in Fig. 6 create and manipulate DYMO messages which are represented by the colour sets defined in Fig. 7. The definition of the colour sets used for modelling the DYMO messages is based on

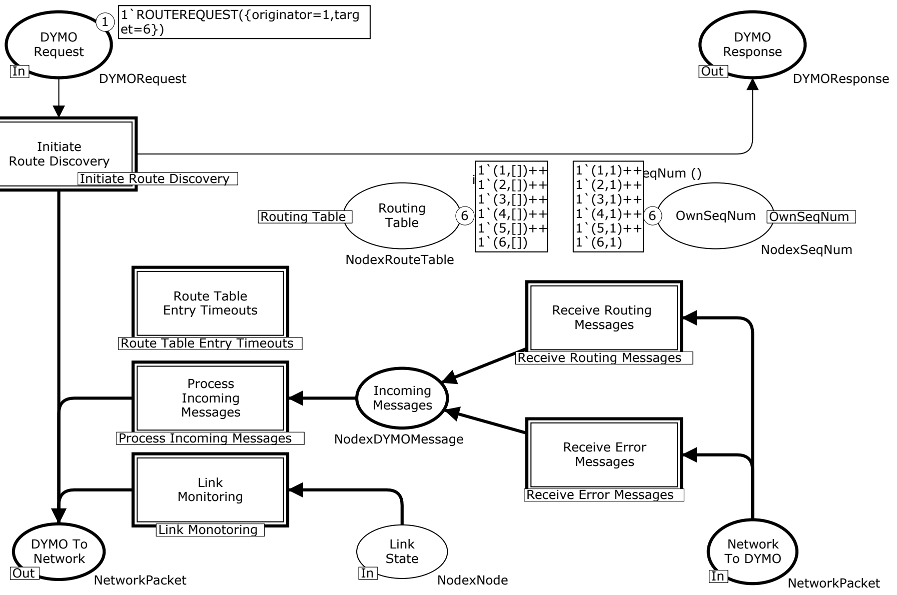


Fig. 6. The DYMOProtocol module.

a direct translation of the description of DYMO messages as found in the DYMO specification [10]. In particular, the same names of message fields as in [10] have been used. The model abstracts from the compact packet layout defined for the DYMO protocol. This is done to ease the readability of the CPN model, and since the packet layout is not important when considering only the functional operation of the DYMO protocol.

The place RoutingTable and the place OwnSeqNum are used to model the routing table and the sequence number of nodes, respectively, that are maintained as part of the internal state of each mobile node. In the marking depicted in Fig. 6 both of these places contain a multi-set containing six tokens. Within the boxes specifying the colours of the individual tokens, ++ (pronounced and) is used to denote union of multi-sets and ‘ (pronounced of) is used to specify the coefficients (i.e., the number of occurrences of tokens with a given colour). The colour set `SeqNum` used to represent the sequence number of a node was defined above, and the colour set `RouteTable` is defined in Fig. 8. To allow each node to have its own sequence number, we use the colour set `NodexSeqNum`. The marking in Fig. 6 corresponds to a MANET with six mobile nodes. The first component of each token on the place OwnSeqNum specifies the identity of a node and the second component specifies the sequence number of the node. Initially, the sequence number of all nodes is set to one. Similarly, it can be seen that the routing table of each mobile node is empty as represented by the empty list (`[]`) specified for each node in the marking of RoutingTable.

```
colset SeqNum          = int with 0 .. 65535;
colset NodexSeqNum     = product Node * SeqNum;
colset NodexSeqNumList = list NodexSeqNum;

colset RERRMessage = record HopLimit         : INT *
                            UnreachableNodes : NodexSeqNumList;

colset RoutingMessage = record TargetAddr : Node   * OrigAddr   : Nodes *
                               OrigSeqNum : SeqNum * HopLimit   : INT   *
                               Dist       : INT;

colset DYMOMessage = union RREQ : RoutingMessage + RREP : RoutingMessage +
                           RERR : RERRMessage;
```

Fig. 7. Colour set declarations for DYMO messages.

The submodules of the DYMOProtocol module all need to access the routing table and the sequence number maintained by each node. To reduce the number of arcs in the modules, the routing table and the sequence numbers have been modelled using *fusion sets*. A fusion set allow a set of places in different modules to linked together into one compound place across the hierarchical structure of the model. In this case, we have a fusion set OwnSeqNum (for linking together places modelling the sequence number of each node) and a fusion RoutingTable (for linking the places modelling the routing table of each node). The name of the fusion set which a place belongs to (if any) is written in a tag positioned next to the place.

Initiate Route Discovery Module. We consider the InitiateRouteDiscovery module shown in Fig. 9 as a representative example of a submodule at the most detailed level of the CPN model. This module specifies how the route discovery procedure is initiated when a request for a route discovery arrives via

```
colset RouteTableEntry = record
                            Address        : IPAddr * SeqNum : SeqNum *
                            NextHopAddress : IPAddr * Broken : BOOL   *
                            Dist           : INT;

colset RouteTable      = list RouteTableEntry;
colset NodexRouteTable = product Node * RouteTable;
```

Fig. 8. Colour set declarations for routing table entries.

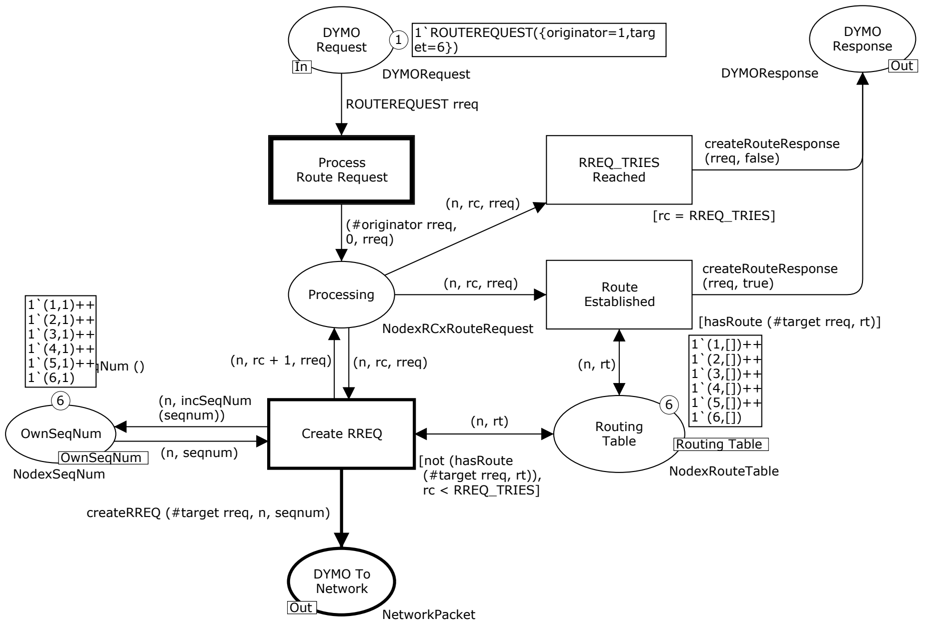


Fig. 9. The Initiate Route Discovery module - initial marking.

the DYMORequest input port. The rectangles in Fig. 9 are ordinary transitions (i.e., non substitution transitions) which means that they can become *enabled* and *occur*. In the marking shown in Fig. 9, a token corresponding to a request for a route discovery originating at node 1 and targeting node 6 is present on the DYMORequest place. In this marking, the transition ProcessRouteRequest is enabled in the following *binding*:

$\langle$`rreq=`$\{$`originator=1,target=1`$\}\rangle$

which binds the *variable* `rreq` of colour set `RouteRequest` to the value in the `ROUTERREQUEST`. Evaluating the input arc expression on the arc from DYMORequest to ProcessRouteRequest results in a multi-set consisting of the single token present on place DYMORequest. The effect of an occurrence of ProcessRequest with the binding above in the marking in Fig. 9 is that the token on DYMORequest is removed and a token is added to place Processing. The colour of the token added to Processing is obtained by evaluating the *arc expression* on the arc from ProcessRouteRequest to Processing in the binding from above:

```
(#originator rreq,0,rreq)
```

The SML operator `#originator` extracts the originator field in the value bound to `rreq`. The marking resulting from the occurrence of ProcessRouteRequest is shown in Fig. 10. A route request being processed is represented by a token on Processing over the colour set `NodexRCxRouteRequest` which is a product type.

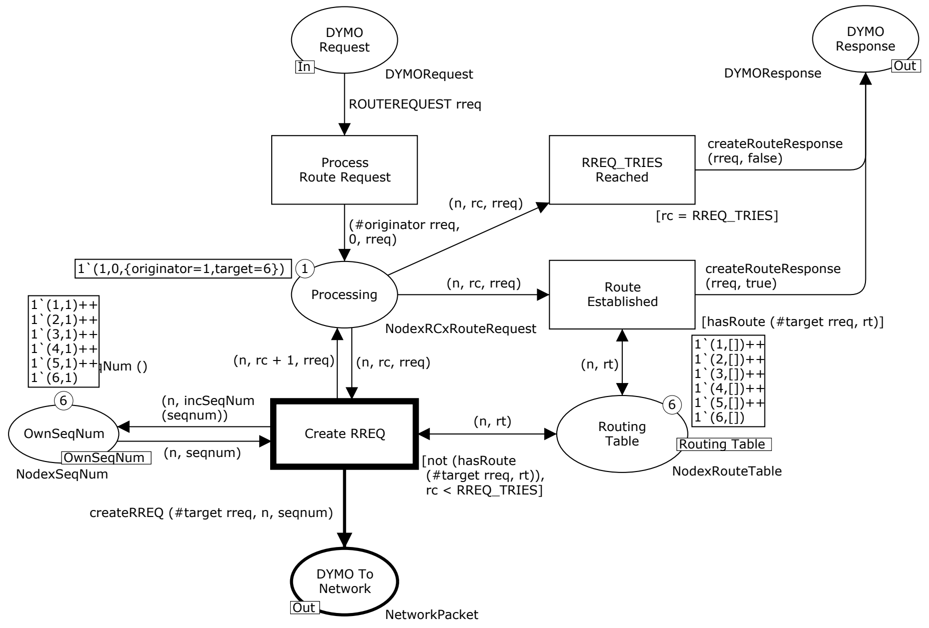


Fig. 10. The Initiate Route Discovery module - after ProcessRouteRequest occurrence.

The first component of the token on Processing specifies the node processing the route request (i.e., the originator), the second component specifies how many times the RREQ has been retransmitted, and the third component specifies the route request.

In the marking shown shown in Fig. 10, the transition CreateRREQ is enabled with the binding:

$\langle$`rc=[],rreq={originator=1,target=1},rc=0,n=1,seqnum=1`$\rangle$

The expression in square brackets positioned next to the CreateRREQ transition is a *guard* specifying an additional boolean conditions (beyond the presence of required tokens on input places) for the CreateRREQ transition to be enabled. In this case, the guard specify that for the transition to enabled, a route must not already exist in the route table `rt` to the target node `#target`, and the number of times `rc` the current route request has been retransmitted must be less than the retransmission limit `RREQ_TRIES` for RREQs. The SML function `hasRoute` used in the guard is implemented as follows:

```
fun hasRoute (target, rt:RouteTable) =
    List.exists (fn {Address, ...} => UNICAST(target) = Address) rt
```

and uses the predefined SML function `List.exists` to check whether an entry in the route table `rt` leading to the `target` node already exists. This is a typical example of how SML is used to represent (sequential) data manipulation.

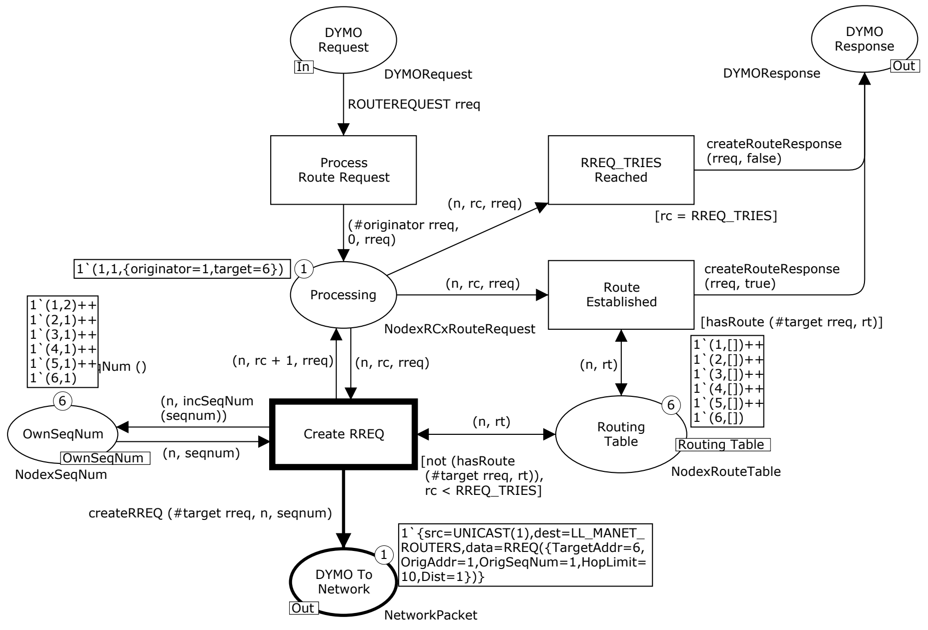


Fig. 11. The Initiate Route Discovery module - after CreateRREQ occurrence.

The marking resulting from occurrence of CreateRREQ is shown in Fig. 11. When sending a RREQ, the sequence number of node 1 sending the request is incremented by 1 and so is the counter specifying how many times the RREQ has been transmitted. Furthermore, a token corresponding to a network packet containing a RREQ message is produced on place DYMOToNetwork. The destination of the packet is set to `LL_MANET_ROUTERS` since it must be sent to all nodes within reach of node 1.

If a route becomes established (i.e., the originator receives a RREP for the RREQ), the RouteEstablished transition becomes enabled and a token can be put on place DYMOResponse indicating that the requested route has been successfully established. If the retransmission limit for RREQs is reached (before a RREP is received), the RREQ_TRIES Reached transition becomes enabled and a token can be put on place DYMOResponse indicating that the requested route could not be established.

2.4 The DYMO Protocol Environment

The MobileWirelessNetwork module shown in Fig. 12 captures the mobile wireless network that DYMO is designed to operate over. It consists of two parts: one part modelling the transmission of network packets represented by the substitution transition WirelessPacketTransmission, and one part representing the mobility of the nodes represented by the Mobility substitution transition. The places DY-

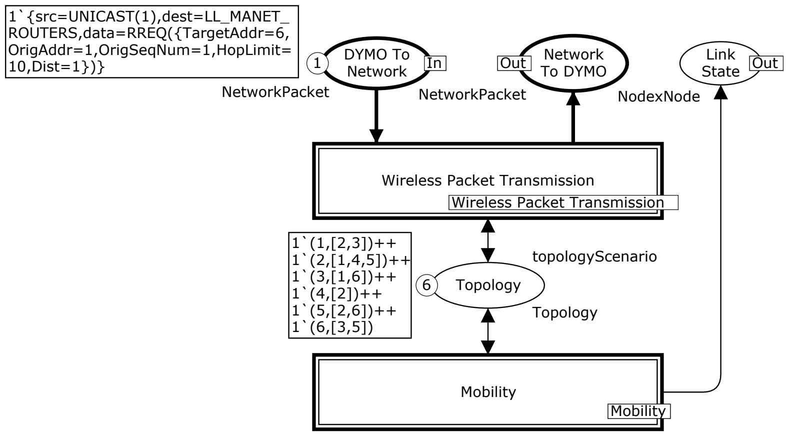


Fig. 12. The Mobile Wireless Network.

MOToNetwork, NetworkToDYMO, and LinkState are associated to the similarly named socket places in Fig. 4. The transmission of network packets is done relative to the current topology of the MANET which are explicitly represented via the current marking of the Topology place. The topology is represented using the colour set `Topology` defined in Fig. 13.

The idea is that each node has an adjacency list of nodes that it can reach in one hop, i.e., its neighbouring nodes. The marking of place Topology in Fig. 12 corresponds to the topology in Fig. 1(left). This adjacency list is then consulted when a network packet is being transmitted from a node to determine the set of nodes that can receive the network packet. In this way, the dynamic topology is modelled by the addition and removal of nodes from the adjacency lists. The place LinkState models that a node can be informed about reachability of its neighbouring nodes which is used in active link monitoring. The CPN model presented in [8] provides a more abstract modelling approach that does not use an explicit representation of MANET topology.

The WirelessPacketTransmission module models the actual transmission of packets and is shown in Fig. 14. The module captures how network packets are transmitted via the physical network from one node to the next. Packets are transmitted over the network according to the function transmit on the arc from the transition Transmit to the place NetworkToDYMO. When the Transmit

```
colset NodeList = list Node;
colset Topology = product Node * NodeList;
```

Fig. 13. Colour set declarations for topology modelling.

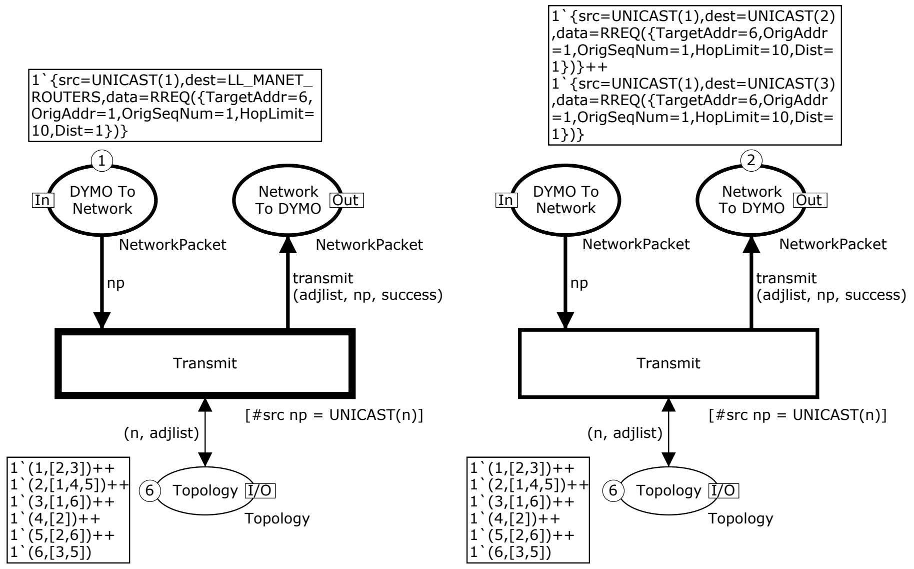

1`{src=UNICAST(1),dest=LL_MANET_ROUTERS,data=RREQ({TargetAddr=6, OrigAddr=1,OrigSeqNum=1,HopLimit=10,Dist=1})}
1
In
DYMO To Network
Network To DYMO
Out
NetworkPacket
np
NetworkPacket
transmit (adjlist, np, success)
Transmit
[#src np = UNICAST(n)]
(n, adjlist)
1`(1,[2,3])++ 1`(2,[1,4,5])++ 1`(3,[1,6])++ 1`(4,[2])++ 1`(5,[2,6])++ 1`(6,[3,5])
6
Topology
I/O
Topology
1`{src=UNICAST(1),dest=UNICAST(2),data=RREQ({TargetAddr=6,OrigAddr=1,OrigSeqNum=1,HopLimit=10,Dist=1})}++
1`{src=UNICAST(1),dest=UNICAST(3),data=RREQ({TargetAddr=6,OrigAddr=1,OrigSeqNum=1,HopLimit=10,Dist=1})}
2

Fig. 14. Transmission of packets - before (left) and after (right) transmission.

transition occurs in a binding where the boolean variable `success` is set to `true`, then all nodes within reach of the sending node will receive the packet. Otherwise, no nodes will receive the packet. The transition Transmit is enabled in the marking shown in Fig. 14 (left) and the marking resulting from a successful transmission of the packet on DYMOToNetwork is shown in Fig. 14 (right). In this case two tokens are added to place NetworkToDYMO corresponding to nodes 2 and 3 receiving the packet being multi-casted from node 1.

In a real network, a transmission could be received by any subset of the neighbouring nodes (e.g., because of signal interference). Here it is only modelled that either all of the neighbouring nodes receives the packet or none receives it. This is sufficient because the modelling of the dynamic topology means that a node can move out of reach of the transmitting node immediately before the transmission occurs which has exactly the same effect as a signal interference in that the node does not receive the packet. Hence, signal interference and similar phenomena imply that a node does not receive a packet is in the model equivalent to the node moving out of reach of the transmitting node.

2.5 Impact of the DYMO CPN Modelling

The development of the DYMO CPN model was based on the natural language specification provided in the Internet draft [9] specifying the DYMO protocol. The modelling work was done when version 10 [9] was the most recent DYMO specification. In the process of constructing the CPN model and simulating it, several issues and ambiguities in the specification were discovered. The most important ones are summarised in Table 2. These issues were submitted to the IETF MANET Working Group mailing list [60] and issue 1 and 3-7 were acknowledged by the DYMO developers and taken into account in the subsequent version DYMO specification [10] (version 11). Issue 2 was not considered critical by the working group as it causes route discovery to fail in scenarios which according to the experience of the DYMO developers seldom would occur in practise.

The modelling conducted with the DYMO protocol illustrates that the construction of a formal and executable model provides a very systematic and comprehensive way of reviewing a protocol design document (such as the DYMO Internet draft) and how it can contribute to increasing the quality of a protocol

Table 2. DYMO CPN modelling [18]: issues identified in the modelling phase.

Issue	Description
1	When processing a routing message, a DYMO router may respond with a RREQ flood, i.e., a RREQ addressed to the node itself, when it is target for a RREQ message (cf. [9], Sect. 5.3.4). It was not clear from the specification which information to put in the RREQ message, i.e., the originator address, hop limit, and sequence number of the RREQ.
2	When judging the usefulness of routing information, the target node is not considered. This means that a new request with a higher sequence number can make an older request for another node stale since the sequence number in the old message is smaller than the sequence number found in the routing table.
3	When creating a RREQ message the distance field in the message is set to zero. This means that for a given node n an entry in the routing table of a node n' connected directly to n may have a distance to n which is 0. Distance is a metric indicating the distance traversed before reaching n, and the distance between two directly connected nodes should be one.
4	In the description of the data structure route table entry (cf. [9], Sect. 4.1) it is suggested that the address field can contain more than one node. It was not clear why this was the case.
5	When processing RERR messages (cf. [9], Sect. 5.5.4) it is not specified whether the hop limit shall be decremented.
6	When retransmitting a RREQ message (cf. [9], Sect. 5.4), it was not explicitly stated whether the node sequence number should be increased.
7	Version 10 of DYMO introduced the concept of distance instead of hop count. Distance is a more general metric, but in the routing message processing (cf. [10], Sect. 5.3.4) it is incremented by one. We believe it should be up to the implementers how much distance is incremented depending on the metric used.

design specification. A number of other issues related to the functionality of the DYMO protocol was reported in [8]. Similar conclusions can also be drawn from other case studies where CPN modelling has been applied to protocols developed in the context of IETF.

3 GAN Architecture and State Space Exploration

The Generic Access Network (GAN) [1] architecture specified by the 3rd Generation Partnership Project (3GPP) [2] allows access to common telephone services such as SMS and voice-calls via IP networks. This section summarises a project [21] in which CPN modelling and state space exploration was used at TietoEnator Denmark. CPNs were applied in the early phases of developing an implementation corresponding to a particular instantiation [31] of the generic GAN architecture [1]. A central part of the GAN architecture is the establishment of a secure connection between a *mobile station* (e.g., a mobile phone) and a *GAN controller* through a *security gateway*. The GAN architecture relies on standardised protocols such as Dynamic Host Configuration Protocol (DHCP) for IP address configuration, IP Security (IPsec) [43] for encryption and authentication, and Internet Key Exchange v2 (IKEv2) protocol [42] for negotiation of IPsec parameters.

The purpose of the CPN model constructed in the project was two-fold. Firstly, to define the scope of the protocol software to be developed by TietoEnator. More specifically, the aim was to determine which parts of the generic GAN specification were to be included in the implementation to be developed by TietoEnator. Secondly, to specify the detailed design and usage of the involved protocol software components. The focus of the CPN model is on the establishment of a secure tunnel and the initial GAN message exchanges since this is where important details were not provided in the full GAN specification. In particular, the full GAN specification [1] contained no clear specification of the IKEv2 message exchange and the states that the protocol entities should be in when establishing a GAN connection (at the time of the project in 2007). Furthermore, the GAN specification only states that IKEv2 and IPSec are to be used, and in which operating modes. The goal of applying state space exploration was to verify the completeness of the design. This included verifying that all phases, steps, and messages involved in establishing a secure GAN connection was covered by the design, and the correctness of the connection establishment (i.e., that a GAN connection is eventually established with the mobile station and the GAN controller being properly synchronised).

3.1 GAN Secure Connection Establishment

The CPN model of the secure connection establishment consists of 31 modules organised into four hierarchical levels. In the following, we present four selected modules from the CPN model with the purpose of illustrating how the phases

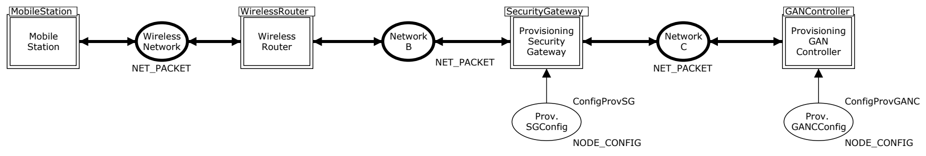


Fig. 15. Top-level module of the GAN model.

that the protocol entities enter when establishing a GAN connection have been modelled. A more in-depth presentation of the CPN model can be found in [21].

Figure 15 shows the top-level module which is organised so that it mimics the GAN network architecture. The substitution transition MobileStation represents the mobile station which is connecting to the telephone network via an IP network. The place Wireless Network connected to MobileStation represents the wireless network which connects the mobile station to a wireless router represented by the substitution transition WirelessRouter. The wireless router is an arbitrary access point with routing functionality, and is connected to the Provisioning Security Gateway, through NetworkB. As part of establishing a GAN connection, an encrypted tunnel is established between the mobile station and the security gateway. The encrypted tunnel is provided by the Encapsulating Security Payload (ESP) mode of the IP security layer (IPSec) [43]. To provide such an encrypted tunnel, both ends have to authenticate each other, and agree on both an encryption algorithm and keys. This is achieved using the Internet Key Exchange v2 (IKEv2) protocol [42]. The provisioning security gateway is connected to the Provisioning GAN Controller via NetworkC. The GAN controllers are connected to the telephone network and performs the relay of traffic to/from the IP networks (NetworkC and the WirelessNetwork). This in turn allows mobile stations to access the services on the telephone network. The places with thin lines connected to the substitution transitions Provisioning Security Gateway and Provisioning GAN Controller are used to provide configuration information to the corresponding network nodes.

Neither the WirelessRouter nor NetworkB are required to be owned or operated by the telephone operator. However, all security gateways, GAN controllers, and NetworkC are assumed to be under the control of the telephone network operator, as non-encrypted messages will be exchanged across them. The CPN model does not include modelling of the telephone network as the scope of the CPN model covers the components involved in establishing the connection with the GAN controller. Furthermore, as the purpose of the model was to represent the protocol entities present on each of the nodes in the network architecture, it sufficed that the model encompass one mobile node, one wireless router, one provisioning security gateway, and one provisioning GAN controller.

The basic exchange of messages in establishing a GAN connection to the provisioning GAN controller involves three steps. The first step is for the mobile station to acquire an IP address on the wireless network using DHCP. The second phase is to create a secure tunnel to the provisioning security gateway. Having established the secure tunnel, the third phase is for the mobile station to open a secure connection to the GAN controller and register itself.

Figure 16 (left) shows the IKEInitiator module of the mobile station and Fig. 16 (right) shows the IKEResponder module of the security gateway. These two peer modules model the second step of the GAN connection establishment concerned with creating the secure tunnel. Incoming IP packets for the module arrives via the ReceiveBuffer input port places. Outgoing IP packets are put in the SendBuffer places. The states (phases) that the protocol entities goes through during the IKE message exchange when establishing the secure tunnel is represented by the places connecting the substitution transitions.

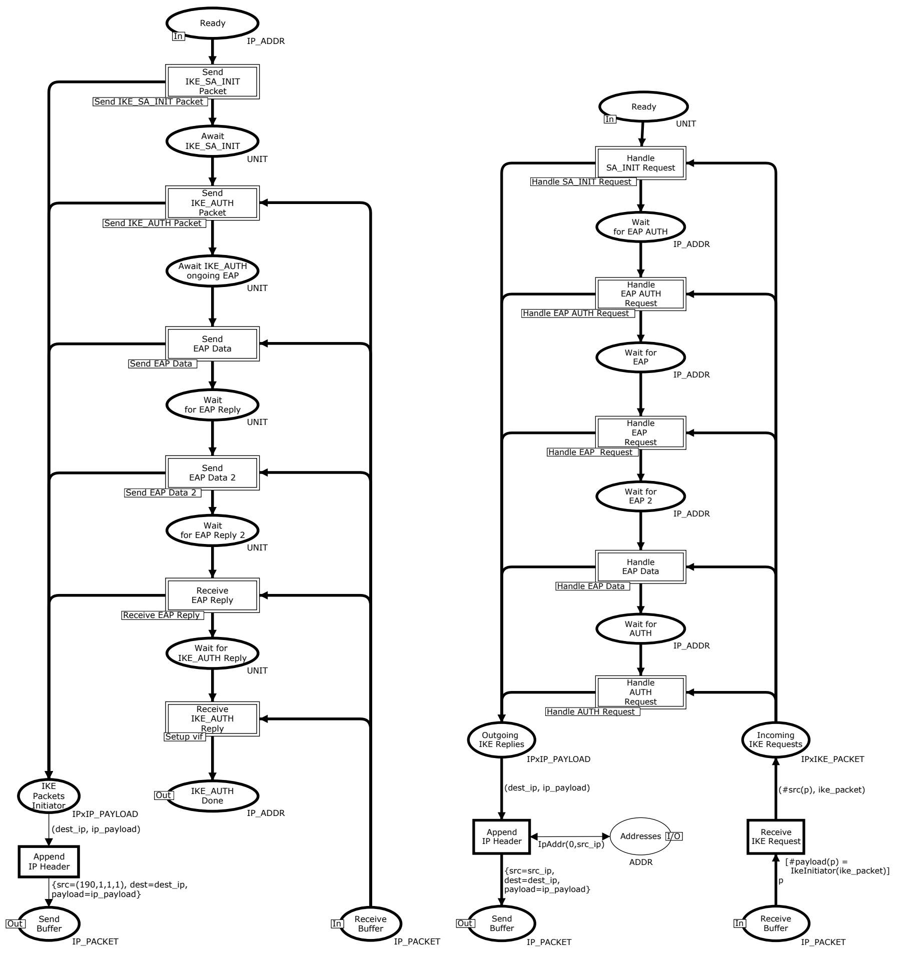
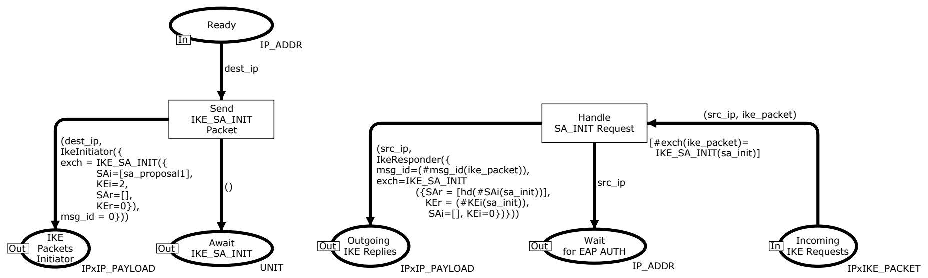


Fig. 16. IKE initiator (left) and IKE responder (right) modules.

The state changes are represented by substitution transitions. The submodules of the substitution transitions specifies the processing rules for messages during the individual phases. Figure 17 shows the Send IKE_SA_INIT Packet and Handle_SA_INIT_Request modules which are the submodules of the two top-most substitution transitions in Fig 16. The Send IKE_SA_INIT Packet transition in Fig. 17 (left) takes the IKE Initiator from the state Ready to Await IKE_SA_INIT and sends an IKE message to the security gateway initialising the communication. The IP address of the security gateway is retrieved from the Ready place. Figure 17 (right) shows how the IKE_SA_INIT packet is handled by the IKE Responder. The guard of the HandleSA_INIT_Request transition ensures that the transition is only enabled if the incoming packet (token) on IncomingIKERequest represent a IKE_SA_INIT packet. In that case, it sends an IKE packet back to the initiator as specified by the arc expression on the arc from Handle_SA_INITRequest and the responder enters the Wait for EAP Auth state. The submodules of the other substitution transitions in Fig. 16 are similar.

The establishment of a GAN connection involves multiple layers of the IP network stack. DHCP (used to configure the mobile station) and GAN are application layer protocols, IKE is a transport layer protocol, and IPSec belongs to the network layer. As a consequence, the CPN model of GAN connection therefore spans multiple protocol layers. Furthermore, the protocol entities also access and manipulate the *routing table* and a *security policy database* (SPD) which is maintained at the IP network layer. The establishment of a GAN connection access the routing table of a node in order to ensures that packets are put into the secure tunnel, and extracted again at the other end. The SPD describes what packets are allowed to be sent and received by the IP protocol stack, and is also responsible for identifying which packets are to be tunnelled at the mobile station and the security gateway. Each entry in the SPD contains the source and destination addresses to use for matching packets, and an action to perform. Modelled actions are *bypass* (which means allow packet to pass without tunnelling) and *tunnel* (the matched packet is to be sent through an ESP tunnel). As we will see later, the content of the routing table and the SPD play an important role in validating the correctness of the GAN connection establishment and. It was therefore required to explicitly represent them in the CPN model.



Ready
In
IP_ADDR
dest_ip
Send IKE_SA_INIT Packet
(dest_ip, IkeInitiator({ exch = IKE_SA_INIT({ SAi=[sa_proposal1], KEi=2, SAr=[], KEr=0}), msg_id = 0}))
()
Out
IKE Packets Initiator
IPxIP_PAYLOAD
Out
Await IKE_SA_INIT
UNIT
Handle SA_INIT Request
(src_ip, ike_packet)
[#exch(ike_packet)= IKE_SA_INIT(sa_init)]
(src_ip, IkeResponder({ msg_id=(#msg_id(ike_packet)), exch=IKE_SA_INIT ({SAr = [hd(#SAi(sa_init))], KEr = (#KEi(sa_init)), SAi=[], KEi=0})}))
src_ip
Out
Outgoing IKE Replies
IPxIP_PAYLOAD
Out
Wait for EAP AUTH
IP_ADDR
In
Incoming IKE Requests
IPxIKE_PACKET

Fig. 17. Example of IKE initiator (left) and IKE responder (right) submodules.

3.2 Verification of the GAN Connection Establishment

Verification of the key properties of the design for secure connection establishment was done by *state space exploration*. The basic idea underlying state space exploration is to compute all reachable states and state changes of the CPN model and represent these as a directed graph, where nodes represent markings and arcs represent occurring binding elements. State spaces can be constructed fully automatically by the state space tool in CPN Tools. Verification of the GAN scenario modelling by means of state spaces relied on the use of the *state space report* that can be generated by CPN Tools. The generation of a state space report for the smallest possible configuration of a considered protocol is typically the first step performed when conducted verification of a CPN model.

The state space report is divided into several sections. In the following we present excepts from the individual sections and explain how they can be used for the verification. Figure 18 shows the first part of the state space report for the CPN model. This part provides some *state space statistics* specifying how large the state space is. It can be seen that the state space consists of $3,854$ nodes and $9,225$ arcs. The construction of the state space took 4 seconds. Statistics for the strongly connected component graph (SCC-graph) are also specified. It has $3,514$ nodes and $8,881$ arcs, and was calculated in 2 seconds. The fact that there are fewer nodes in the SCC-graph than in the state space implies that there are non-trivial strongly connected components (SCCs). A non-trivial SCC is an SCC consisting of more than a single state space node and hence contain a cycle of the state space. This means that infinite executions exist and that the GAN connection establishment may not terminate. We will investigate the reasons for this at the end of this subsection.

The *boundedness properties* section of the state space report specifies how many and which tokens a place may hold – when considering all reachable states (markings). Figure 19 lists the best upper and best lower integer bounds for selected places in the mobile station module. It can be seen that the first four places modelling the states of the mobile station contains at most one token and may contain zero tokens. Similarly, it can be seen that there is at most one token in the send, received, and network buffers. The place RoutingTable has a lower integer bound of 0 and and upper integer bound of 1. The lower integer bound is 0 since in the initial marking there are no tokens on this place. During the start-up procedure of the mobile station, a token representing a list of routing table

```
State Space                         Scc Graph
  Nodes:   3,854                      Nodes:   3,514
  Arcs:    9,225                      Arcs:    8,881
  Secs:        4                      Secs:        2
```

Fig. 18. State space report – statistics.

```
Best Integer Bounds        Upper      Lower
   Down                     1          0
   Ready                    1          0
   VIF open to Prov. SG     1          0
   VIF Closed               1          0

   Send Buffer              1          0
   Receive Buffer           1          0
   Network Buffer           1          0

   Routing Table            1          0
   Security Policy Database 1          1
```

Fig. 19. State space report – integer bounds.

entries is put on this place. The place SecurityPolicyDatabase has a best upper and a best lower integer bound of 1. This means that there is always exactly one token present on this place. This is because the security policy database is modelled as a single token being a list containing the current entries in the security policy database.

The *best upper multi-set bound* of a place specifies for each colour in the colour set of the place the maximal number of tokens that is present on this place with the given colour in any reachable marking. This is specified as a multi-set, where the coefficient of each value is the maximal number of tokens with the given value. If the coefficient is zero, then the colour is omitted in the specification. Figure 20 shows part of the state space report providing the upper multi-set bounds for the security policy databases of the mobile station, wireless router, security gateway, and the GAN controller. The upper multi-set bounds specifies the possible tokens that can reside on these places and by carefully inspecting these bounds it was possible to validate that the possible entries in the security policy database were all as desired. Altogether inspection of the boundedness properties helped significantly in increasing confidence in the correctness of the design in terms of proper settings of the routing table and the security policy database.

Figure 21 shows the part of the state space report specifying the *home and liveness properties*. The home properties show that there exists a single *home marking*, which is state number 3854. A home marking is a state which can be reached from any reachable state. For the GAN scenario model this means that it is impossible to have an execution sequence starting from the initial state (initial marking) which cannot be extended to reach state 3854. The liveness properties tell us that there is a single *dead marking* which is also state number 3854. A dead marking is a state in which no transitions are enabled. This means that the marking corresponding to node 3854 is both a home and a dead mark-

```
Mobile Station: Security Policy Database
    1`[{src=((0,0,0,0),0),dest=((0,0,0,0),0),
        nl_info=PayloadList([PAYLOAD_DHCP]),policy=SpdBypass}]++
    1`[{src=((80,1,1,1),32),dest=((12,0,0,0),8),
        nl_info=AnyNextLayer,policy=ESPTunnel(((190,1,1,1),(172,1,1,2)))},
       {src=((190,1,1,1),0),dest=((0,0,0,0),0),
        nl_info=AnyNextLayer,policy=SpdBypass}]++
    1`[{src=((190,1,1,1),0),dest=((0,0,0,0),0),
        nl_info=AnyNextLayer,policy=SpdBypass}]

Wireless Router: Security Policy_Database
    1`[{src=((0,0,0,0),0),dest=((0,0,0,0),0),
        nl_info=AnyNextLayer,policy=SpdBypass}]

Security Gateway: Security Policy Database
    1`[{src=((13,0,0,0),8),dest=((80,1,1,1),32),
        nl_info=AnyNextLayer,policy=ESPTunnel(((172,1,1,2),(190,1,1,1)))},
       {src=((0,0,0,0),0),dest=((0,0,0,0),0),
        nl_info=AnyNextLayer,policy=SpdBypass}]

GAN Controller: Security Policy Database
    1`[{src=((0,0,0,0),0),dest=((0,0,0,0),0),
        nl_info=AnyNextLayer,policy=SpdBypass}]
```

Fig. 20. State space report – best upper multi-set bounds.

ing. To obtain information above the marking corresponding to node number 3854, it was transferred into the simulator of CPN Tools and displayed graphically on the CPN model. It was then checked (by inspecting the markings of the individual places) that the marking corresponded to the desired terminating state of the GAN connection establishment procedure, i.e., the state where the mobile station has obtained an IP address, has successfully communicated with the provisioning GAN controller, all protocol modules are in a state corresponding to the GAN connection having been established, and the routing tables and security databases contain the correct entries. The fact that state 3854 is the only dead marking tells us that the protocol as specified by the CPN model is partially correct, i.e., if execution terminates we have the correct result. Furthermore, because node 3854 is also a home marking it is always possible to terminate the GAN connection establishment with the correct result.

The analysis above showed that it is always possible to terminate the GAN connection establishment procedure correctly, but it does not guarantee that it will eventually happen. The section of the state space report providing information about *fairness properties* showed that the two transitions RejectDiscoveryRequest and HandleGARCReject which are part of the GAN controller module were

```
Home Properties                 Liveness Properties
  Home Markings: [3854]           Dead Markings: [3854]
```

Fig. 21. State space report – home and liveness properties.

impartial. This means that these two transitions occur infinitely often in any infinite occurrence sequence. The two transition occurs if the GAN controller decides to reject an incoming connection from a mobile station. Hence, if the connection establishment procedure does not terminate in the single home and dead marking identified, then it is because the GAN controller keeps rejecting the connection.

3.3 Perspective on GAN Architecture Validation

The validation of secure connection establishment in the considered GAN scenario is representative for how validation of protocols are typically performed with CPN Tools – as it in practise involves a combination of both simulation and state space exploration. As part of the construction of the GAN model, the support for *interactive simulation* in CPN Tools was used to perform detailed checks to ensure that the model behaviour was as desired. An interactive simulation is similar to single-step debugging. During an interactive simulation, the next step is manually selected between the enabled bindings in the current marking, and it is possible to observe the effects of the individual steps directly on the graphical representation of the CPN model. Even though the use of interactive simulations (and simulation in general) cannot be used to prove correct behaviour, it proved to be very useful in identifying situations related to improper manipulations of the entries in the routing tables and security policy database - or where additional detail not present in the GAN specification had to be worked out and specified. Furthermore, interactive simulation was helpful in identifying issues that led the GAN connection establishment procedure to terminate prematurely, e.g., because a certain phase of the connection establishment was missing in the CPN model. These issues manifested themselves in markings where the GAN connection had not yet been established, but where no transitions were enabled. This was in particular effective in making explicit where further specification of the message exchanges were required.

The interactive simulation was in later phases replaced with *automatic simulation* where a number of random executions of the CPN model were performed with the purpose of checking whether the execution of the CPN model resulted in state in which the GAN connection was properly established. Eventually state space exploration of the CPN model was conducted which succeeded in establishing the key property that a GAN connection will eventually be established provided that the GAN controller does not keep rejecting the connection request. The verification conducted also illustrated the general observation that in many case, the use of basic state space exploration and the state space report (i.e., investigating standard behavioural properties of Petri nets) are sufficient in establishing key properties of a protocol design. In this case the state space was small in size and could be generated in a very few seconds without the use of advanced state space exploration techniques.

4 Interoperability Protocol and Behavioural Visualisation

During a protocol model construction phase it is common to use interactive, single-step simulation of the CPN model to investigate the operation of the protocol in detail as was discussed in the context of GAN connection establishment in the previous section. In this case, the execution of the CPN model is viewed directly on its graphical representation and provides a simple way of validating that the model operates as intended. Even though the CPN modelling language supports abstraction and hierarchical modules there can still be a significant amount of detail being presented with this approach, and observing every single step in a simulation is often too detailed a level of observation when investigating the behaviour of a model. Furthermore, even if the CPN model is executable, it still lacks the application- and domain-specific appeal of a conventional software prototype.

This section summarises a project conducted at Ericsson Telebit A/S addressing the specification of the Routing Interoperability Protocol (RIP)[3] for routing packets between IP core networks and mobile ad-hoc networks. The main purpose of the routing interoperability protocol is to ensure that a packet flow between a host in the core network and a mobile node in an ad-hoc network is always relayed via one of the closest gateways that connect the core network and the mobile ad-hoc network. In this project, the constructed CPN model was augmented with high-level, application-specific behavioural visualisation reflecting the execution the CPN model using concepts specific to the considered network architecture and RIP. In this way it was possible to investigate the behaviour of the protocol design while overcoming the limitations of interactive simulations. The CPN model augmented with behavioural visualisation was used as an early model-based prototype of RIP. It allowed the protocol design to be discussed among protocol engineers unfamiliar with CPNs, and it also enabled the protocol design to be presented to customers with the purpose of soliciting requirements to the services to be provided by the protocol. The complete CPN model of the RIP protocol is hierarchically structured into 18 modules. An detailed presentation of the CPN model can be found in [51].

4.1 Modelling the Routing Interoperability Protocol

Figure 22 shows the top level module of the CPN model which reflects the network architecture that the RIP protocol is designed to operate in. The network

[3] RIP as discussed in this section should not be confused with the Routing Information Protocol[62]

architecture consists of three parts: an IPv6 core network represented by the CoreNetwork substitution transition (left) and its submodules, a mobile ad-hoc network represented by the AdHocNetwork substitution transition (right) and its submodules, and two gateways represented by the substitution transitions Gateway1 and Gateway2. The basic idea in the interoperability protocol is that the mobile nodes register the IPv6 address in the Domain Name Server (DNS) server in the core network that corresponds to IPv6 address prefix announced by the closest (preferred) gateway. Updates to the DNS database managed by the DNS server relies on the Dynamic Domain Name System Protocol [82].

The places CoreNetwork and AdHocNetwork are used for modelling the packets in transit on the core network and ad-hoc network, respectively. Figure 22 depicts a state in which there is one token on place CoreNetwork and two tokens on place AdHocNetwork. As an example, place CoreNetwork contains one token with the colour:

```
(RECEIVE("3ffe:100:3:401::1"), {src="3ffe:100:3:401::2",
       dest="3ffe:100:3:401::1",cont=DNSREQ("AHN(3)")})
```

representing a DNS request (`DNSREQ`) in transit on the core network from a host with source IPv6 address `3ffe:100:3:401::2` to a DNS server with destination IPv6 address `3ffe:100:3:401::1`. IPv6 addresses are 128-bit and by convention written in hexadecimal notation in groups of 16-bits separated by colon (:). Leading zeros are skipped within each group and a double colon (::) is a shorthand for a sequence of zeros. Addresses consists of an *address prefix* and an *interface identifier*.

The place AdHocNetwork contains two tokens representing gateway advertisements in transit to nodes in the ad-hoc network. The gateways periodically announce their presence to nodes in the mobile ad-hoc network by sending *gateway advertisements* containing an IPv6 *address prefix*. The two Config places

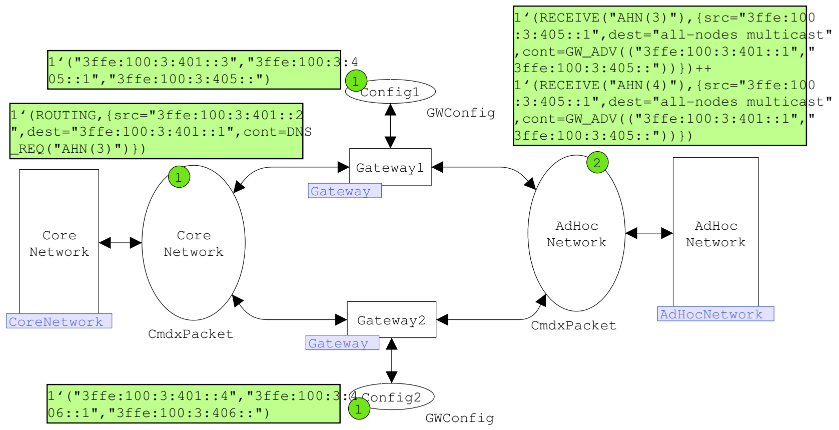

1‘("3ffe:100:3:401::3","3ffe:100:3:405::1","3ffe:100:3:405::")
1
Config1
GWConfig
1‘(ROUTING,{src="3ffe:100:3:401::2",dest="3ffe:100:3:401::1",cont=DNS_REQ("AHN(3)")})
1‘(RECEIVE("AHN(3)"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})++
1‘(RECEIVE("AHN(4)"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})
Gateway1
Gateway
1
2
Core Network
Core Network
AdHoc Network
AdHoc Network
CoreNetwork
CmdxPacket
Gateway2
Gateway
CmdxPacket
AdHocNetwork
1‘("3ffe:100:3:401::4","3ffe:100:3:406::1","3ffe:100:3:406::")
Config2
1
GWConfig

Fig. 22. The System module – top-level module of the CPN model.

contain a token representing the configuration of the corresponding gateway. It consists of the IPv6 address of the gateway interface connected to the core network, the IPv6 address of the gateway interface connected to the ad-hoc network, and the address prefix announced by the gateway. Address prefixes are written on the form x/y where x is an IPv6 address and y is the length of the prefix. The mobile nodes in the ad-hoc network configure IPv6 addresses based on the received gateway advertisements. In the marking depicted in Fig. 22, Gateway1 is announcing the 64-bit address prefix 3ffe:100:3:405::/64 and Gateway2 is announcing the prefix 3ffe:100:4:406::/64. Each of the gateways has configured an address on the interface to the ad-hoc network based on the prefix they are announcing to the ad-hoc network. Gateway1 has configured the address 3ffe:100:3:405::1 and Gateway2 has configured the address 3ffe:100:3:406::1. The gateways have also configured addresses on the interface to the core network based on the 3ffe:100:3:401::/64 prefix of the core network.

Figure 23 lists the definitions of the colour sets used in the System module. IP addresses, prefixes, and symbolic IP addresses are represented by colour sets IPAdr, Prefix, and Symname all defined as the set of strings. The colour set PacketCont and Packet are used for modelling the IP packets. The five different kinds of packets used in RIP are modelled by the PacketCont colour set:

`DNS_REQ` modelling a DNS request packet. It contains the symbolic IP address to be resolved to a (numerical) IP address by a DNS server.

`DNS_REP` modelling a DNS reply. It contains the symbolic IP address and the resolved IP address.

`DNS_UPD` modelling a DNS update. It contains the symbolic IP address to be updated and the new IP address to be bound to the symbolic address.

`GW_ADV` modelling the advertisements disseminated from the gateways. An advertisement contains the IP address of the gateway and the announced prefix.

`PACKET` modelling generic payload packets belonging to packet flows between hosts and the mobile nodes.

The colour set `Packet` models the packets as a record containing the source, destination, and content. The actual payload (content) and layout of packets are not essential for modelling the interoperability protocol and has therefore been abstracted away. The colour set Cmd is used to control the operation of the various modules in the CPN model. The colour set GWConfig models the configuration information of the gateway.

The Core Network. Figure 24 shows the CoreNetwork module modelling the core network. This module is the immediate submodule of the substitution transition CoreNetwork of the System module shown in Fig. 22. The port place CoreNetwork is assigned to the CoreNetwork socket place in the System module (see Fig. 22). The substitution transition Routing represents the routing mechanism in the core network. The substitution transition Host represents the host on the core network, and the substitution transition DNS Server represents the DNS server that maintains the DNS database.

```
colset Prefix = string;  (* address prefixes *)
colset IPAdr = string;   (* IP addresses     *)
colset SymName = string; (* symbolic names   *)

colset SymNamexIPAdr = product SymName * IPAdr;
colset IPAdrxPrefix = product IPAdr * Prefix;

colset PacketCont = union DNS_REQ : SymName +       (* DNS Request     *)
                          DNS_REP : SymNamexIPAdr + (* DNS Reply       *)
                          DNS_UPD : SymNamexIPAdr + (* DNS Update      *)
                          GW_ADV  : IPAdrxPrefix +  (* Advertisments   *)
                          PACKET;                   (* Generic payload *)

colset Packet = record src  : IPAdr  * dest : IPAdr * cont : PacketCont;
colset Cmd    = union ROUTING              + RECEIVE      : IPAdr +
                      FLOODING     : IPAdr + GWAHNROUTING : IPAdr +
                      AHNGWROUTING : IPAdr;

colset CmdxPacket = product Cmd * Packet;
colset GWConfig   = product IPAdr * IPAdr * Prefix;
```

Fig. 23. Colour set definitions used in the System module.

The Mobile Ad-hoc Network. Figure 25 depicts the AdHocNetwork module modelling the mobile ad-hoc network. The place Nodes is used to represent the nodes in the mobile ad-hoc network. The place RoutingInformation is used to represent the routing information in the ad-hoc network which is assumed to be available via some routing protocol executed in the ad-hoc network. This routing information enables among other things the nodes to determine the distance to the reachable gateways. Detailed information about the colour of the token on place RoutingInformation has been omitted.

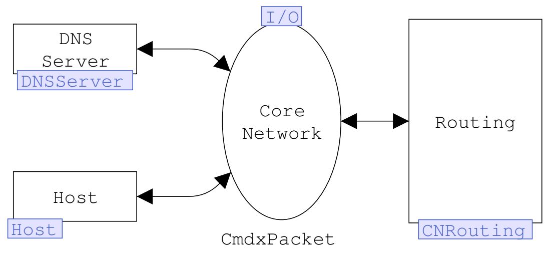


Fig. 24. Core Network module – modelling the core network.

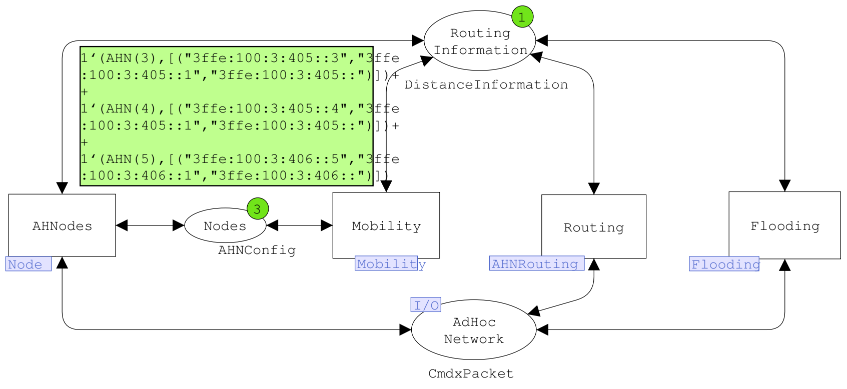

Routing Information
1
DistanceInformation
1`(AHN(3),[("3ffe:100:3:405::3","3ffe:100:3:405::1","3ffe:100:3:405::")])+
1`(AHN(4),[("3ffe:100:3:405::4","3ffe:100:3:405::1","3ffe:100:3:405::")])+
1`(AHN(5),[("3ffe:100:3:406::5","3ffe:100:3:406::1","3ffe:100:3:406::")])
AHNodes
Nodes
3
AHNConfig
Mobility
Routing
Flooding
Node
Mobility
AHNRouting
Flooding
I/O
AdHoc Network
CmdxPacket

Fig. 25. AdHocNetwork module – modelling the ad-hoc network.

Figure 26 lists the definition of the colour sets used in the AdHocNetwork module. The colour set AHNConfig is used to model the configuration information for the mobile ad-hoc nodes. Each ad-hoc node is represented by a token on place Nodes and the colour of the tokens specifies the name of the node and a list of configured IP addresses. Each configuration specifies the IP address configured, and the IP address and prefix of the corresponding gateway. It is possible for a mobile ad-hoc node to configure an IP address for multiple gateways. The mobile node must ensure that the DNS database always contains the IP address corresponding to the *preferred gateway*. In the marking shown in Fig. 25, it can be seen from the labels below the mobile nodes that Ad-hoc Node 3 and Ad-hoc Node 4 have configured IP addresses based on the prefix announced by Gateway1, whereas Ad-Hoc Node 5 has configured an IP address based on the prefix announced by Gateway2. For an example, Ad-hoc Node 3 has configured the address 3ffe:100:3:405::3.

```
(* --- ad-hoc nodes --- *)
color AHId = int with 1..5;
color AHNode = union AHN : AHId;

(* --- configuration information for ad-hoc nodes --- *)
color AHNIPConfig = product IPAdr * IPAdr * Prefix;
color AHNIPConfigs = list AHNIPConfig;
color AHNConfig = product AHNode * AHNIPConfigs;
```

Fig. 26. Colour definitions used in the AdHocNetwork module.

There are four substitution transitions in the AdHocNetwork module corresponding to the components of the ad-hoc network. The substitution transition AHNodes represents the behaviour of the nodes in the mobile ad-hoc network. The substitution transition Mobility models the mobility of nodes in the ad-hoc network, i.e., that the nodes may move closer or further away from the gateways. The substitution transition Routing represents the routing protocol executed in the ad-hoc network. The purpose of the routing protocol in the context of the RIP protocol is to provide the nodes with information about distances to the gateways. The routing is abstractly modelled in a similar way as the routing mechanism in the core network and will not be discussed further in this paper. The substitution transition Flooding models the dissemination of advertisements from the gateways. A detailed presentation of this part of the model has been omitted here but can be found in [51].

4.2 Behavioural Visualisation

CPN Tools can use the BRITNeY Suite animation package [85] to create application-specific graphics on top of CPN models. The animation package is a stand-alone application, and CPN Tools invokes the primitives of the animation package using remote procedure calls. The animation package supports a wide range of diagram types via a plug-in architecture. The reader is referred to [85] for details. In the routing interoperability project, the animation package was used to create an *animation GUI* on top of the CPN model. The animation GUI allow a user to observe the execution of the constructed CPN model using a graphical representation of the network architecture. The graphics is updated by the underlying CPN model according to the execution of the formally specified protocol, and the CPN model is also able to react to stimuli provided by the user via the animation GUI.

Figure 27 shows a representative snapshot of the application-specific graphics during the execution of the CPN model. The IP addresses configured by the individual nodes are shown as labels below the nodes. For an example, Ad-hoc Node 3 has configured two IP addresses: 3ffe:100:3:405:3 and 3ffe:100:3:406:3. The convention is that the preferred IP address is the topmost address in the list below the node. The entries in the DNS database are shown in the upper left corner. It shows the entries for each of the three ad-hoc nodes. The two numbers written at the top of each node are counters that provide information about the number of packets on the incoming (left) and outgoing (right) interfaces of the nodes. Transmissions of advertisements from the gateways are visualised by green dots. Fig. 27 shows an example where Gateway2 is transmitting an advertisement. Transmission of payload packets is visualised using read dots, and DNS packets are visualised using blue dots.

In addition to observing feedback on the execution of the CPN model in the animation GUI, it is also possible to provide input to the CPN model directly via the animation GUI. The user can move the nodes in the ad-hoc network thereby changing the distances to the two gateways. It is also possible to define a packet flow from the host in the core network to one of the nodes in the mobile

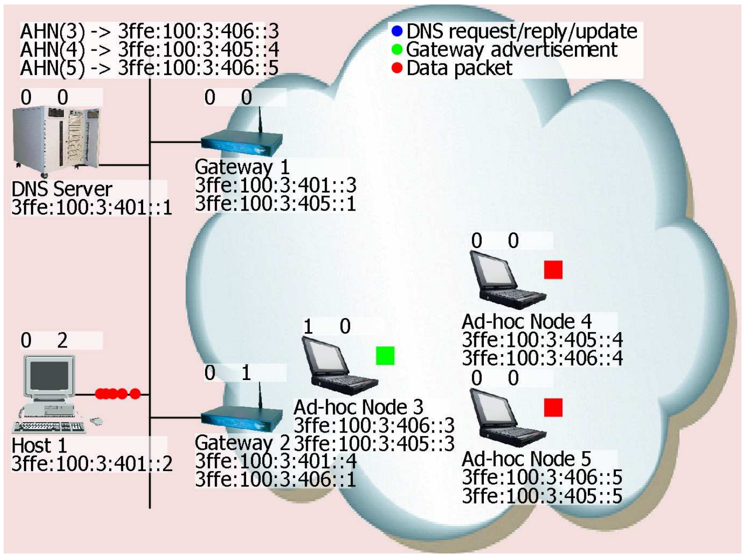


Fig. 27. Snapshot of the interaction graphics.

ad-hoc network by clicking on the red square positioned next to each of the ad-hoc nodes. The square will change its colour to green once the CPN model has registered the flow. The flow can be stopped again by clicking on the (now green) square next to the mobile ad-hoc node. Finally, it is possible to force the transmission of an advertisement from a gateway by clicking on the gateway.

A more generic form of high-level graphical feedback in the form of MSCs was also used in this project. Figure 28 shows an example of an MSC diagram creating based on a simulation of the CPN model. The MSC shows a scenario where Ad-hoc Node 3 makes a Move and discovers that Gateway 2 is now the closest gateway. This causes it to send a DNS update to the DNS server. The last part of the MSC shows the host initiating a packet flow to Ad-hoc Node 3. One benefit of using MSCs is that they provide an event-based view that records the execution history. This is in contrast to the state-based view on the CPN model that one obtains during an interactive simulation. The two forms of feedback therefore complements each other and MSCs have been widely used in projects where CPNs have been applied to protocol design.

Graphical feedback from the execution of the CPN model is achieved by attaching *code segments* to the transitions in the CPN model. These code segments are sequential pieces of code that are executed whenever the corresponding transition occurs in the simulation/execution of the CPN model. As an example consider the CNRouting module in Fig. 29. The transition Route models the routing of the packet on the core network. It uses the routing information on place RoutingInformation to direct the packet to the proper gateway. The SML function FindNextHop in the guard expression of the transition computes the IP

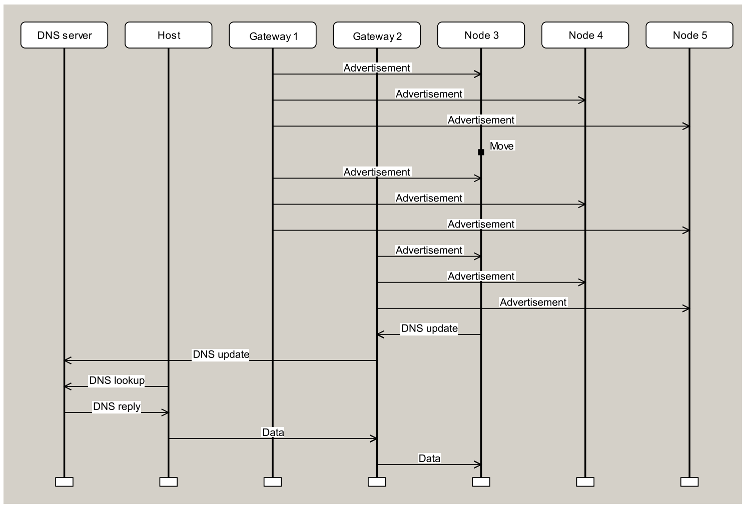
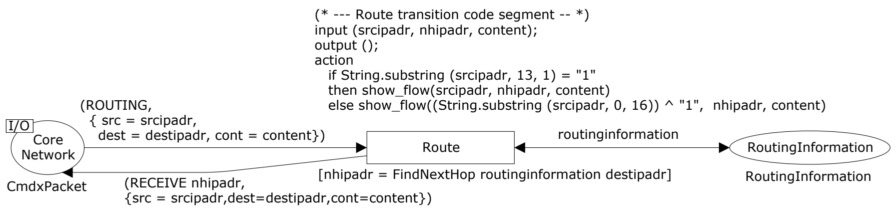


Fig. 28. Message sequence chart generated by the animation GUI.

address of the next hop gateway using the routing information and destination IP address of the packet. The Route transition has an attached code segment which is executed whenever the transition occurs. The code segment invokes the primitives in the animation package for animating the transmission of packets in the core network.

The CPN model receives input from the animation GUI by polling the animation GUI for events. An event queue has been implemented between the animation GUI and the CPN model. The code segment of transition Produce in the Poll module shown in Fig. 30 polls the animation GUI for events at regular intervals during the execution of the CPN model. Events are put into a list-token representing an event queue on the place Events. The parts of the CPN model that is to react on events from the animation GUI are linked via place fusion to the Event place and is able to consume events from the event queue. The occurrence of the transition Produce corresponds to a poll to the animation GUI for events.



```
(* --- Route transition code segment -- *)
input (srcipadr, nhipadr, content);
output ();
action
   if String.substring (srcipadr, 13, 1) = "1"
   then show_flow(srcipadr, nhipadr, content)
   else show_flow((String.substring (srcipadr, 0, 16)) ^ "1",  nhipadr, content)
```

```
(ROUTING,
  { src = srcipadr,
    dest = destipadr, cont = content})
```

```
(RECEIVE nhipadr,
{src = srcipadr,dest=destipadr,cont=content})
```

```
[nhipadr = FindNextHop routinginformation destipadr]
```

Fig. 29. The CNRouting module – Routing in the core network.

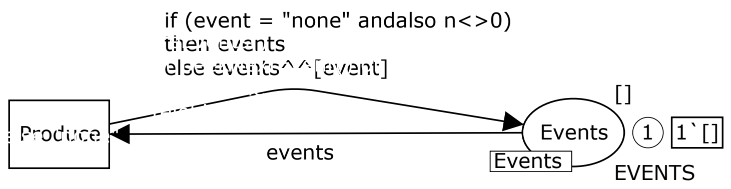


Fig. 30. The Poll module – Polling the animation GUI for events.

4.3 On the Applicability of Behavioural Visualisation

The CPN model combined with the animation GUI that was developed in the RIP project served as an early model-based executable prototype. The domain specific graphical user interface (the animation GUI) made it possible to explore and demonstrate the design of the interoperability protocol with the underlying formal model being transparent for the observer and the demonstrator. In particular, it made it possible for persons without knowledge of the CPN modelling language to experiment with the proposed design. The use of an animation GUI on top of the CPN model has the advantage that the behaviour observed by the user is as defined by the underlying model that formally specifies the design. The alternative would have been to implement a separate visualisation application totally detached from the CPN model. This would have led to double representation of the dynamics of the interoperability protocol which could in turn lead to inconsistencies between the two representation of the design.

Another advantage offered by the development of a model-based prototype is ease of control compared to a physical prototype, in particular in the case of mobile nodes and wireless communication where scenarios can be very difficult to control and reproduce. The use of a model means that there is no need to invest in physical equipment and there is no need to setup the actual physical equipment early in the project. The use of a model also makes it possible to investigate larger scenarios, e.g., scenarios that may not be feasible to investigate with the available physical equipment. Another general advantage of the approach taken in the RIP project is that at an early stage of development, the implementation details can be abstracted away and only the key part of the design have to be specified in detail. As an example, the CPN model of the interoperability protocol abstracted away the routing mechanisms in the core and ad-hoc networks, and the mechanism used for distribution of advertisements. Instead, the service assumed from these components for the interoperability protocol to work was modelled. The possibility of making abstraction means that it is possible to obtain an executable prototype without implementing all the components.

5 Design of an Edge Router Discovery Protocol

The previous sections have demonstrated how modelling, simulation, state space exploration, and behavioural visualisation can be applied for validating the functional design of protocols. This section summarises a project [45] conducted with Ericsson Telebit A/S where a combination of the techniques introduced in the previous sections were applied for the design of an Edge Router Discovery Protocol (ERDP).

The CPN model of ERDP was developed in close cooperation with the protocol engineers at Ericsson Telebit A/S based upon a natural-language specification that would normally have served as a basis for the implementation of the protocol. Simulation and MSCs were used in initial investigations of the ERDP protocol behaviour. Then state space exploration was used to conduct a formal verification of the key behavioural properties of ERDP. The modelling, simulation, and subsequent state space exploration all helped to identify several omissions and behavioural errors in the design, demonstrating the benefits of using the techniques discussed in this paper in a protocol design process.

5.1 ERDP Modelling

ERDP is based on the IPv6 Neighbour Discovery Protocol (NDP) [64] and supports *edge routers* residing on the boundary of an *IP core network* in configuring *gateways* with an IPv6 address prefix. This address prefix can in turn be used by mobile nodes in ad-hoc networks to configure global IPv6 unicast addresses and obtain Internet access via the core network. Figure 31 shows the ERDP module which is the top-level module of the CPN model. The substitution transition Gateway represents the gateway, and the substitution transition EdgeRouter represents the edge router. The wireless communication link between the edge router and the gateway is represented by the substitution transition GW_ER_Link. The four socket places GWIn, GWOut, ERIn, and EROut model packet buffers between the link layer and the gateway and edge router. Both the gateway (GW) and the edge router (ER) have an incoming and an outgoing packet buffer.

All four places in Fig. 31 have the colour set IPv6Packet, used to model the IPv6 packets exchanged between the edge routers and gateways. Since ERDP is based on the IPv6 Neighbour Discovery Protocol, the packets are carried as Internet Control Message Protocol (ICMP) packets. The definitions of the colour sets for NDP, ICMP, and IPv6 packets were derived directly from RFC 2460 [15] by using record type constructors for representing fields within packets and union type constructors for representing the different kinds of packets (see [45] for detail). It was considered important by the protocol engineers for later implementation that the definition of the packets followed closely the structure of IPv6 packets instead of a more abstract representation.

Figure 32 shows the EdgeRouter module. The port places ERIn and EROut are related to the accordingly named socket places in the ERDP module (see Fig. 31). The place Config models the configuration information associated with the edge router, and the place PrefixCount models the number of prefixes still available in the edge router for distribution to gateways. The place PrefixAssigned is used to keep track of which prefixes are assigned to which gateways.

Figure 33 shows the declarations of the colour sets for the three places in Fig. 32. The configuration information for the edge router (modelled by the colour set ERConfig) is a record consisting of the IPv6 link-local address and the link-layer address of the edge router. A list of pairs (colour set ERPrefixAssigned) consisting of a link-local address and a prefix is used to keep track of which prefixes are assigned to which gateways. A counter modelled by the place PrefixCount with the colour set PrefixCount is used to keep track of the number of prefixes still available. When this counter reaches 0, the edge router has no further prefixes available for distribution. The number of available prefixes can be modified by changing the initial marking of the place PrefixCount, which is set to 1 by default.

The substitution transition SendUnsolicitedRA (in Fig. 32) corresponds to the multicasting of periodic *unsolicited router advertisements* (RAs) by the edge router such that gateways can discover the presence of the edge router. When a gateway receives an unsolicited RA, it responds with a unicast *router solicitation* (RS). The substitution transition ProcessRS models the reception at the edge router of unicasted RSs from gateways, and the sending of a unicast RA to the gateway in response. The substitution transition ERDiscardPrefixes models the expiration of prefixes on the edge router side.

The marking shown in Fig. 32 has a single token on each of the three places used to model the internal state of the edge router protocol entity. In the marking shown, the token on the place PrefixAssigned with the colour `[]` corresponds to the edge router not having assigned any prefixes to the gateways. The token on the place PrefixCount with colour `1` indicates that the edge router has a single prefix available for distribution. Finally, the colour of the token on the place Config specifies the link-local and link addresses of the edge router. In this case

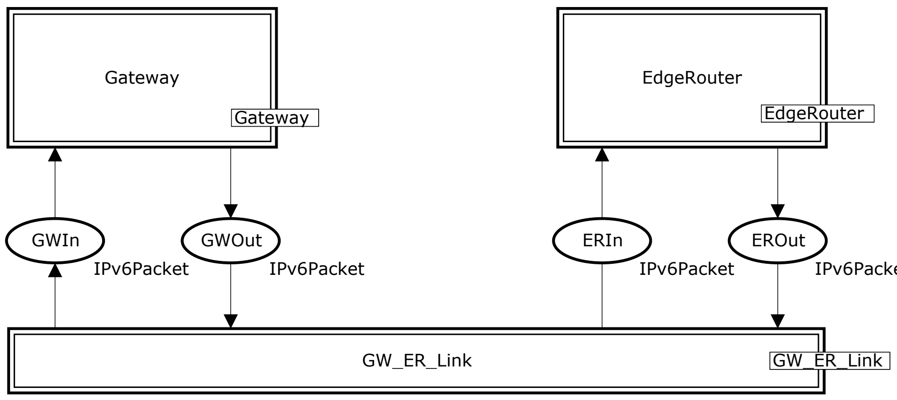


Fig. 31. Top-level module of the ERDP CPN model.

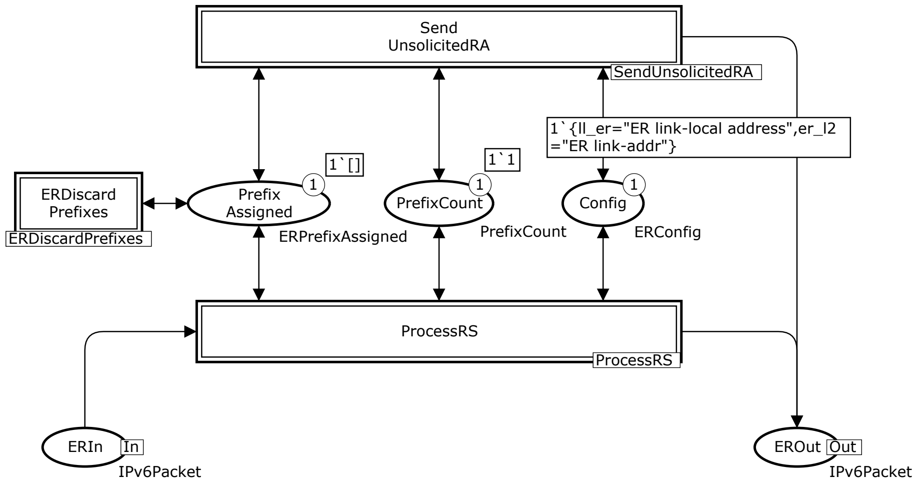


Fig. 32. The EdgeRouter module.

```
colset LinkAddr    = string;

colset ERConfig = record ll_er : IPv6Addr * (* link-local address  *)
                         er_l2 : LinkAddr;  (* link-addr (layer 2) *)

colset ERPrefixEntry    = product IPv6Addr * IPv6Prefix;
colset ERPrefixAssigned = list ERPrefixEntry;

colset PrefixCount = int;
```

Fig. 33. Colour set definitions for edge routers.

the edge router has the symbolic link-local address ER link-local address, and the symbolic link-address ER link-addr.

Figure 34 depicts the SendUnsolicitedRA module which is the submodule of the substitution transition SendUnsolicitedRA in Fig. 32. The transition SendUnsolicitedRA models the sending of the periodic unsolicited router advertisements. The variable erconfig is of type ERConfig, and the variable prefixleft is of type PrefixCount. The transition SendUnsolicitedRA is enabled only if the edge router has prefixes available for distribution, i.e., prefixleft is greater than 0. This is ensured by the function SendUnsolicitedRA in the guard of the transition.

Figure 35 depicts the marking of the SendUnsolicitedRA module after the occurrence of the transition SendUnsolicitedRA in the marking shown in Fig. 34. An unsolicited router advertisement has been put in the outgoing buffer of the edge router. It can be seen that the `DestinationAddress` is the address

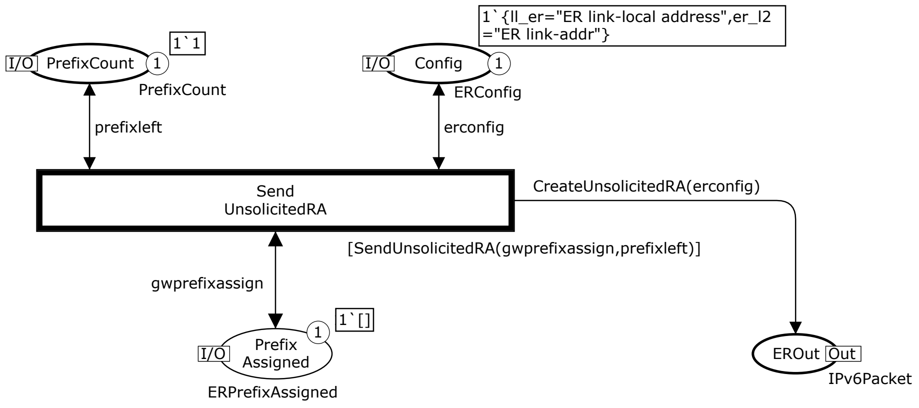
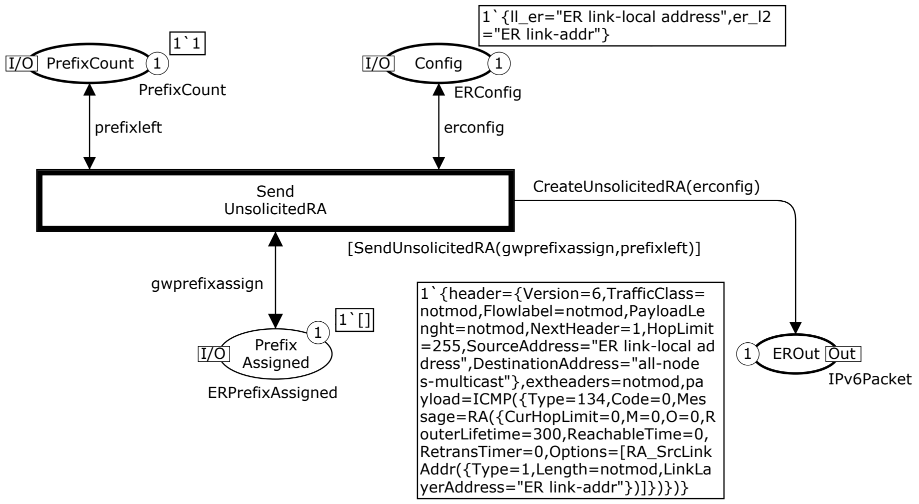


Fig. 34. Initial marking of the SendUnsolicitedRA module.

`all-nodes-multicast`, the `SourceAddress` is `ER link-local address`, and the `LinkLayerAddress` (in the options part) is `ER link-addr`.

Figure 36 shows the part of the GW_ER_Link module that models transmission of packets from the edge router to the gateway across the wireless link. Transmission of packets from the gateway to the edge router is modelled similarly. The port places GWIn and EROut are linked to the similarly named socket places in Fig. 31. The transition ERtoGW models the successful transmission of packets, whereas the transition LossERtoGW models the loss of packets. The variable ipv6packet is of type IPv6Packet. A successful transmission of a packet



```
1`{header={Version=6,TrafficClass=
notmod,Flowlabel=notmod,PayloadLe
nght=notmod,NextHeader=1,HopLimit
=255,SourceAddress="ER link-local ad
dress",DestinationAddress="all-node
s-multicast"},extheaders=notmod,pa
yload=ICMP({Type=134,Code=0,Mes
sage=RA({CurHopLimit=0,M=0,O=0,R
outerLifetime=300,ReachableTime=0,
RetransTimer=0,Options=[RA_SrcLink
Addr({Type=1,Length=notmod,LinkLa
yerAddress="ER link-addr"})]})})}
```

Fig. 35. Module SendUnsolicitedRA, after occurrence of SendUnsolicitedRA.

from the edge router to the gateway corresponds to moving the token modelling the packet from the place EROut to GWIn. If the packet is lost, the token will only be removed from the place EROut.

Wireless links, in general, have a lower bandwidth and higher error rate than wired links. These characteristics have been abstracted away in the CPN model since the purpose is to reason not about the performance of ERDP but rather its logical correctness. Duplication and reordering of messages are not possible on typical one-hop wireless links, since the detection of duplicates and the preservation of order are handled by the data-link layer. The modelling of the wireless links does allow overtaking of packets, but this overtaking was eliminated in the state space exploration phase where bounds were imposed on the capacity of the input and output packet buffers.

The CPN model was developed as an integrated part of the development of ERDP. The creation of the CPN model was done in cooperation with the protocol engineers at Ericsson in parallel with the development of the ERDP specification. Altogether 70 person-hours were spent on CPN modelling. Prior to the development of the CPN model, the protocol engineers at Ericsson were given a 6 hour course on CPNs that made the capable of reading CPN models. This means that CPN models could be used actively in discussion related to the design of the ERDP protocol. MSCs (to be illustrated shortly), integrated with simulation were used in both review steps to investigate the behaviour of ERDP in detail. The use of MSCs in the project was of particular relevance since it presented the operation of the protocol in a form well known to the protocol engineers. Altogether 24 design issues (as discussed in Sect. 1 of this paper) were identified during three iterations on the CPN model.

5.2 ERDP Protocol Verification

State space exploration was conducted after the three iterations of modelling as outlined above. The purpose of the state space exploration was to conduct a more thorough investigation of the operation of ERDP, including verification of its key properties. One important property of ERDP is proper configuration of the gateway with prefixes. This means that for a given prefix and state where the gateway has not yet been configured with that prefix, the protocol must be

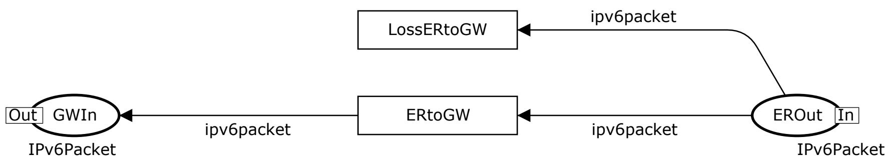


Fig. 36. Part of the GW_ER_Link module.

able to configure the gateway with that prefix. Furthermore, when the gateway has been configured with the prefix, the edge router and the gateway should be *consistently configured*, i.e., the assignment of the prefix must be recorded both in the gateway and in the edge router protocol entity. Whether a marking represents a consistently configured state for a given prefix can be checked by inspecting the marking of the place PrefixAssigned in the edge router and the marking of the place Prefixes in the gateway.

The first step towards state space exploration of the CPN model was to obtain a finite state space. The CPN model as presented above has an infinite state space, since an arbitrary number of tokens (packets) can be put on the places modelling the packet buffers. As an example, the edge router may initially send an arbitrary number of unsolicited router advertisements. To obtain a finite state space, an upper integer bound of 1 was imposed on each of the places GWIn, GWOut, ERIn, and EROut (see Fig. 31) which model the packet buffers. This also prevents overtaking among the packets transmitted across the wireless link. Furthermore, the number of packets simultaneously present in the four input/output buffers was limited to 2. Technically, this was done by using the *branching options* available in the CPN state space tool to prevent the processing of enabled transitions whose occurrence in a given marking would violate the bounds the imposed bounds on the buffer places.

The second step was to consider the simplest possible configurations of ERDP, starting with a single prefix and assuming that there is no packet loss on the wireless link and that prefixes do not expire. The full state space for this configuration had 46 nodes and 65 arcs. Inspection of the state space report showed that there was a single dead marking represented by node 36. Inspection of this node showed that it represented a state where all of the packet buffers were empty. However, the edge router and gateway were inconsistently configured in this state in that the edge router had assigned the prefix P1 (the single prefix), while the gateway was not configured with that prefix. This was an error in the protocol. To locate the source of the problem, query functions in the state space tool were used to obtain a counterexample leading from the node representing the initial marking to node 36. Figure 37 shows the resulting error trace, visualised by means of an MSC. This MSC was generated automatically from the extracted counter example. The column labelled GWBuffer represents the packet buffer between the gateway protocol entity and the underlying protocol layers. Similarly, the ERBuffer column represents the packet buffer in the edge router. The problem is that the edge router sends two unsolicited RAs. The first one gets through and the gateway is configured with the prefix, which can be seen from the event marked with *A* in the lower part of the MSC. However, when the second RS, without any prefixes, is received by the edge router (the event marked with *B*), the corresponding solicited RA will not contain any prefixes. Because of the way the protocol was specified, the gateway will therefore update its list of prefixes to the empty list (the event marked with *C*), and the gateway is no longer configured with a prefix.

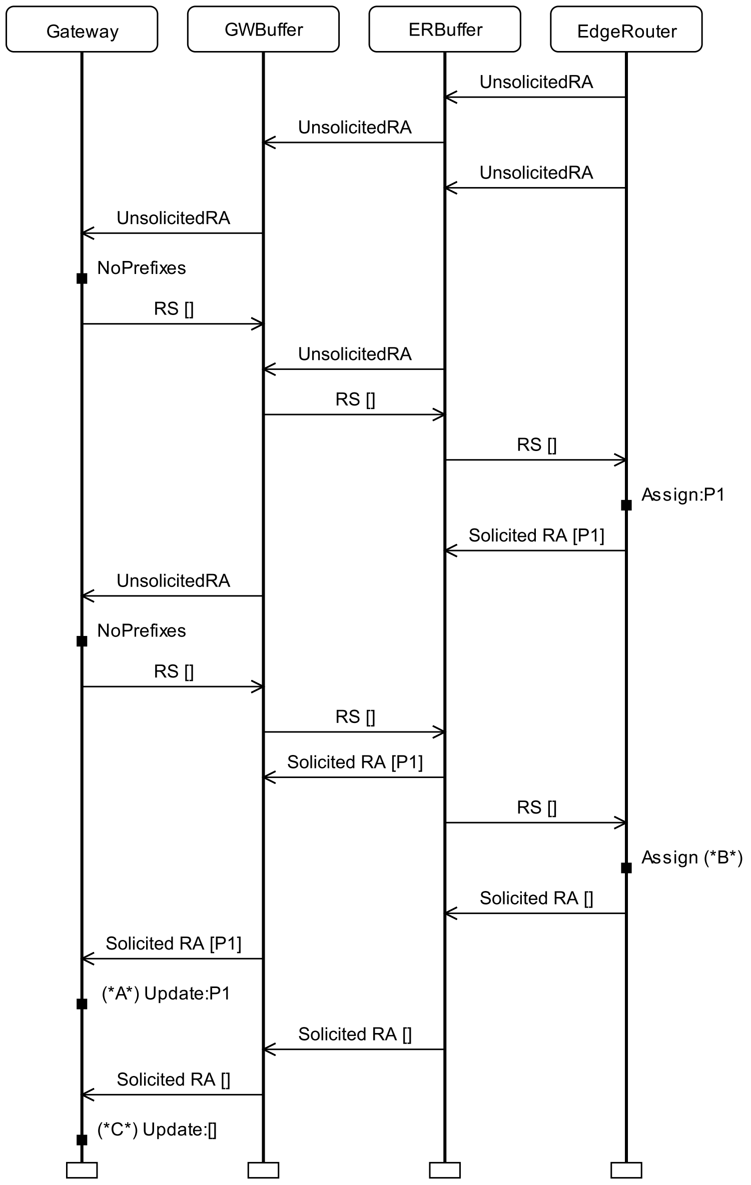


Fig. 37. MSC showing an execution leading to an undesired terminal state.

To fix the error, the protocol was modified so that the edge router always replies with the list of all prefixes that it has currently assigned to the gateway. The state space for the modified protocol consisted of 34 nodes and 49 arcs, and there were no dead markings in the state space. The state space report specified that there were 11 home markings. Inspection of these 11 markings showed that they all represented consistently configured states for the prefix P1. The markings were contained in the single terminal SCC of the state space. A terminal SCC is an SCC of the state space where all successors of states in the SCC belong to the SCC itself. This shows that, from the initial marking it is always possible to reach a consistently configured state for the prefix, and that when such a marking has been reached, the protocol entities will remain in a consistently configured state. To verify that a consistently configured state would eventually be reached, it was checked that the single terminal SCC was the only non-trivial SCC. A trivial SCC is a SCC consisting of just a single state. This showed that all cycles in the state space (which correspond to non-terminating executions of the protocol) were contained in the single terminal SCC, which (from above) contained only consistently configured states. The reason why the protocol is not supposed to terminate in a consistently configured state represented by a dead marking is that the gateway may, at any time, when it is configured, send a router solicitation back to the edge router to have its prefixes refreshed.

When the correctness of the protocol had been established for a single prefix, the number of prefixes was increased. When there is more than one prefix available it no longer holds that a marking will *eventually* be reached where *all* prefixes are consistently configured. The reason is that with more than one prefix, the edge router may at any time decide not to configure the gateway with additional prefixes. Hence, a state where all prefixes have been consistently configured might not eventually be reached. Instead, firstly, it was verified that there was a single terminal SCC, all markings of which represent states where all prefixes have been consistently configured. This shows that it is always possible to reach such a marking, and when the protocol has consistently configured all prefixes, the protocol entities will remain consistently configured. Secondly, it was checked that all markings in each non-trivial SCC represented markings where the protocol entities were consistently configured with a subset of the prefixes available in the edge router. The properties above was checked using a number of user-defined queries in the state space tool of CPN Tools.

The third step was to allow packet loss on the wireless link between the edge router and the gateway. First, the case was considered in which there is only a single prefix for distribution. The state space for this configuration had 40 nodes and 81 arcs. Inspection of the state space report showed that there was a single dead marking. This marking represented an undesired terminal state where the prefix had been assigned by the edge router, but the gateway was not configured with the prefix. The source of the problem was located by extracting a counter examples and visualising it in a similar manner as shown in Fig. 37. The problem was fixed by ensuring that the edge router would resend an unsolicited RA to the gateway as long as it had prefixes assigned to the gateway. The state space of the revised CPN model had 68 nodes and 160 arcs. Inspection of the state space report showed that there were no dead markings and no home markings. Investigation of the terminal SCCs showed that there were two terminal SCCs, each containing 20 markings. The nodes in one of them all represented states where the edge router and gateway were consistently configured with the single prefix P1, whereas the nodes in the other terminal SCC all represented states where the protocol entities were not consistently configured. The markings in the undesired terminal SCC represent a livelock in the protocol, i.e., if one of the markings in the undesired terminal SCC is reached, it is no longer possible to reach a state where the protocol entities are consistently configured with the prefix. The source of the livelock was related to the control fields used in the router advertisements for refreshing prefixes and their interpretation on the gateway. This was identified by obtaining the MSC for a path leading from the initial marking to one of the markings in the undesired terminal SCC. As a result, the processing of router advertisements in the gateway was modified. The state space for the protocol with the modified processing of router advertisements also had 68 nodes and 160 arcs. The state space had a single terminal SCC containing 20 nodes, which all represented states where the protocol entities were consistently configured with the single prefix.

When packet loss is present, it is not immediately possible to prove that the two protocol entities will eventually be consistently configured. The reason is that any number of packets can be lost on the wireless link. Each of the non-trivial SCCs was inspected using a user-defined query to investigate the circumstances under which the protocol entities would not eventually be consistently configured. This query checked that either all nodes in the non-trivial SCC represented consistently configured states or none of the nodes in the SCC represented a consistently configured state. For those non-trivial SCCs where no node represented a consistently configured state, it was checked that all cycles contained the occurrence of a transition corresponding to loss of a packet. Since this was the case, it can be concluded that any failure to reach a consistently configured states will be due to packet loss only. Hence, if finitely many packets are lost, a consistently configured state for some prefix will *eventually* be reached.

The fourth and final step in the analysis was to allow prefixes to expire. The analysis was conducted first for a configuration where the edge router had only a single prefix to distribute. The state space for this configuration had 173 nodes and 513 arcs. The state space had a single dead marking, and inspection of this dead marking showed that it represented a state where the edge router has no further prefixes to distribute, it has no prefixes recorded for the gateway, and the gateway is not configured with any prefix. This marking is a desired terminating state of the protocol, as we expect prefixes to eventually expire. Since the edge router has only finitely many prefixes to distribute, the protocol should eventually terminate in such a state. The single dead marking was also a home marking, meaning that the protocol can always enter the expected terminal state.

When prefixes can expire, it is possible that the two protocol entities may never enter a consistently configured state. The reason is that a prefix may expire in the edge router (although this is unlikely) before the gateway has been successfully configured with that prefix. Hence, we are only able to prove that for any marking where a prefix is still available in the edge router, it is possible to reach a marking where the gateway and the edge router are consistently configured with that prefix.

Table 3 lists statistics for the size of the state space in the three verification steps for different numbers of prefixes. The column '|P|' specifies the number of prefixes. The columns 'Nodes' and 'Arcs' give the numbers of nodes and arcs, respectively, in the state space. For the state spaces obtained in the first verification step, it can be seen that 38 markings and 72 arcs are added for each additional prefix. The reason for this is that ERDP proceeds in phases where the edge router assigns prefixes to the gateway one at a time. Configuring the gateway with an additional prefix follows exactly the same procedure as that for the assignment of the first prefix. Once the state space had been generated, the verification of properties could be done in a few seconds. It is also worth observing that as the assumptions are relaxed, i.e., we move from one verification step to the next, the sizes of the state spaces grow. This, combined with the identification of errors in the protocol even in the simplest configuration, without packet loss and without expiration of prefixes, shows the benefit of starting state space exploration from the simplest configuration and gradually lifting assumptions. Furthermore, the state explosion problem was not encountered during the verification of ERDP, and the key properties of ERDP were verified for the number of prefixes that were envisioned to appear in practise.

Table 3. State space statistics for the three verification steps.

	No loss/No expire		Loss/No Expire		Loss/Expire	
|P|	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs
1	34	49	68	160	173	531
2	72	121	172	425	714	2 404
3	110	193	337	851	2 147	7 562
4	148	265	582	1 489	5 390	19 516
5	186	337	926	2 390	11 907	43 976
6	224	409	1 388	3 605	23 905	89 654
7	262	481	1 987	5 185	44 550	169 169
8	300	553	2 742	7 181	78 211	300 072
9	338	625	3 672	9 644	130 732	505 992
10	376	697	4 796	12 625	209 732	817 903

5.3 Conclusions from the ERDP Project

The construction of a CPN model and subsequent state space exploration can be seen as a very thorough and systematic way of reviewing the ERDP design specification. The project showed that the process of constructing a CPN model based on the ERDP specification provided valuable input to the ERDP design, and the use of simulation added further insight into the operation of the protocol. State space exploration, starting with the simplest possible configuration of the protocol, identified additional errors in the protocol. The results from state space exploration also demonstrate that errors are often present in the smallest configurations of a protocol system.

Using an iterative process where both a conventional natural-language specification and a CPN model were developed (as in this project) turned out to be an effective way of integrating CPN modelling and validation into the development of a protocol. In general, the combination of an executable formal model (such as a CPN model) and a natural-language specification seems to be provide a useful way to develop a protocol. One reason why both are required is that the software engineers that are eventually going to implement the protocol (which may be different from those that design the protocol) in many cases will not be familiar with the CPN modelling language. Secondly, in many cases there are important implementation elements of the protocol specification that are not reflected in the CPN model, such as the layout of packets.

It can be argued whether or not the issues and errors discovered in the process of modelling and conducting state space exploration would have been identified if additional conventional reviews of the ERDP specification had been conducted. Some of them probably would have been, but more subtle problems such as the inconsistent configurations discovered during state space exploration would probably not have been discovered until the first implementation of ERDP was operational. The reason for this is that discovering these problems requires one to consider subtle execution sequences of the protocol.

Overall, the application of CPNs in the development of ERDP was considered a success for three main reasons. Firstly, it was demonstrated that the CPN modelling language and supporting computer tools were powerful enough to specify and verify a real-world protocol being developed in an industrial project, and that integration into the conventional protocol development process is not difficult. Secondly, the act of constructing the CPN model, executing it, and discussing it led to the identification of several non-trivial design errors and issues that, under normal circumstances, would not have been discovered until, at best, the implementation phase. Finally, the effort of constructing the CPN model and conducting state space exploration was represented by approximately 100 person-hours. This is a relatively small investment compared with the many issues that were identified and resolved early as a consequence of constructing and analysing the CPN model.

6 Summary and Perspectives

Functional validation of protocol designs is one of the main application areas of CPNs and supporting computer tools [20]. In this paper, we have surveyed a selection of recent projects on modelling and functional validation of industry relevant protocols. The examples demonstrate how the elements of protocols can be modelled using CPNs, and they illustrate how a combination of simulation, application-specific behavioural visualisation, and state space exploration is typically applied in protocol validation with CPNs. From a modelling perspective, the protocol examples have ranged from models representing two (or few) peer protocol entities (e.g., GAN, EDRP, and RIP) having an explicit representation in the net structure, to parameterised models capable of modelling an arbitrary number of peer protocol entities by setting a model parameter (e.g., DYMO). The latter was based on constructing a folded model where the identity of the protocol entities is encoded explicitly as part of the token colours. The CPN models have also illustrated modelling at different protocols layer ranging from models operating at a single protocol layer (e.g., DYMO and ERDP) to protocol system design involving multiple protocol layers and protocols (e.g., GAN and RIP). An important aspect of the examples is that the process of modelling and conducting single step simulation is an important (but often underestimated) activity in the validation of a protocol design.

6.1 State Space Exploration

The main technique available for functional verification of CPN models is that of explicit state space exploration. The examples presented in this paper show how basic state space exploration combined with the generation of a state space report relying on a number of standard behavioural properties of Petri Nets, provides a light-weight approach which in many cases is an important step in verifying key properties of a protocol design. The main reason for the wide spread application of state space exploration has been the presence of mature computer tool support combined with the main advantages of state space exploration in terms of being a highly systematic approach, being able to provide counter examples, and allowing for a high degree of automation. State space exploration of CPN models has been supported by several generations of CPN computer tools. It was supported by the early stand-alone Occurrence Graph Analyser (OGA), the state space tools integrated in Design/CPN [12] and later CPN Tools [70], and also the more recent ASAP model checking platform [83].

Advanced state space methods aimed at alleviating the inherent state explosion problem [77] have been a highly active area of research, and resulted in a vast variety of storage techniques and verification algorithms. Early work on addressing state explosion in the context of CPNs concentrated on computer tool support for, and initial experiments with, the equivalence [36], symmetry [14, 17, 34, 37], and the stubborn set methods [76]. The symmetry and equivalence methods rely on constructing a condensed state space where each node represents an equivalence class of states and each arc represents an equivalence class of events. The symmetry method has, e.g., been applied on a mutual exclusion protocol [40] and an embedded systems protocol [58]. The practical experiments showed that the time required to compute canonical representatives of equivalence classes was excessive – even with the use of advanced group algebraic techniques [59]. The equivalence method has only been used on a small stop-and-wait protocol [41] due to the obligation of providing a manual soundness proof for the user-provided equivalence relation. The stubborn set method [75, 77] relies on analysing enabling and disabling dependencies between transitions (binding elements) and use this to explore only a subset of the enabled transitions (binding elements) in each state encountered during state space exploration. The rich inscription language (i.e., Standard ML) which is fundamental building block of the CPN modelling language, however, pose problems for the analysis of binding element dependencies in the context of CPNs [50] – unless relying on an unfolding of the CPN model to the equivalent Place/Transition net. Hence, restrictions on the modelling language is required to apply the stubborn set method without relying on unfolding. Introducing restrictions to the CPN modelling language is a research direction which has been explored only to a limited extent. Another widely used verification approach in the context of CPNs is based is the methodology of [5]. A central component of this approach is an explicit modelling of both the protocol and its service, and the use of finite-state automata language comparison as a criteria for checking that the protocol conforms to the specified service.

Recent work on addressing the state explosion problem in the context of CPNs has concentrated on making more economical use of memory resources when exploring the state space. Memory is (in many cases) the limiting factor in state space exploration of CPN models due to the large state vectors. This work resulted in the development of the sweep-line method [13] and the comback method [19, 84]. The sweep-line suite of methods [4, 13, 46, 47, 61] is aimed at on-the-fly verification and exploits a notion of progress found in many concurrent systems. Exploiting progress allows for the deletion of states from memory during a progress-first traversal of the state space. This in turn reduces peak memory usage. The comback method can be viewed as an exploration-order independent storage mechanism based on hash compaction [72, 86]. It allows the usually large state vectors of CPN models to be stored in compact form, and the full state vector of a state is reconstructed when needed for comparison with newly generated states. Unlike the classical hash compaction method, the comback method guarantees full coverage of the state space. The ASAP model checking platform has support for a number of these advanced state space methods – including methods developed outside the context of CPNs.

The compact modelling of protocols enabled by CPNs have in many cases had the effect that the full state space can be explored for at least the smallest configuration of the considered protocol. The GAN and ERDP examples presented in this paper are concrete examples illustrating this. Practise have shown that the primary capability offered by the advanced state space methods is the possibility of verifying larger configurations of the protocol - and in some cases [49] the configurations of the system that are expected to occur in practise. The ERDP example considered in this paper is another example of this. Hence, despite that explicit state space exploration methods requires one to conduct verification relative to a particular configuration of the protocol, the current suite of availably state space methods combined with the power of modern computing platforms in many situations allows for the practical validation of industrial-sized protocols. Examples of protocols for which parametric verification has been pursued in the context of CPNs can be found in [22, 23].

6.2 Further Examples of Protocol Validation

The four protocol examples discussed in this paper constitute only a small subset of the examples that have been published in the literature - in particular related to protocols developed in the context of IETF and other protocol standardisation bodies.

The Datagram Congestion Control Protocol (DCCP) developed by the IETF has been investigated in [7]. DCCP is intended to provide an unreliable transport service with congestion control mechanisms. The work in [7] was done in parallel with the development of the emerging DCCP standard, and concentrated on modelling and verification of the connection establishment and synchronisation procedures of DCCP. It resulted in the identification of several functional errors in the protocol design, including discovery of deadlocks, non-progress behaviour (chatter), and problems with connection establishment in relation to sequence number wraps. The formal validation resulted in the IETF working group making small (but important) changes to the connection establishment and synchronisation procedures of DCCP. The work also included the development of a formal service specification for DCCP [24] and application of the sweep-line method [78] for on-the-fly checking of the protocol conformance to the developed service specification.

The classical Transmission Control Protocol (TCP) has also been modelled and verified using CPNs [6]. Similar to the work on DCCP, this work concentrated on the connection establishment procedures. It resulted in verifying the absence of dead and livelocks in connection establishment, and a detailed specification of the circumstances under which TCP connection establishment may not be successful. Another example of transport layer protocol modelling and validation can be found in [79] which considers the Stream Transmission Control Protocol (SCTP).

The Internet Open Trading Protocol (IOTP) designed to provide an interoperability framework for Internet commerce was formally modelled and validated using CPNs in [66–68]. IOTP is designed to handle common trading procedures and encompass trading roles such as consumer, merchant, payment handler, and delivery handler. IOTP is organised around a collection of eight baseline transactions consisting of Purchase, Refund, Value exchange, Authentication, Withdrawal, Deposit, Inquiry, and Ping. These transactions comprise a minimal set of transactions for an Internet commerce protocol. A formal specification of the service provided by IOTP was developed using CPN in [68]. The service was specified in the form of a finite-state automaton labelled with service primitives. The automaton was extracted from the state space of the CPN model by identifying the binding elements corresponding to service primitives of the protocol. A CPN model of the IOTP protocol itself was presented in [66, 67]. State space exploration focussed on the termination properties and absence of livelocks in the IOTP transactions. The use of state space exploration revealed deficiencies related to termination of transactions. A verification of the IOTP protocol CPN model [66, 67] against the formal service specification from [68] was presented in [65]. Finite-state automate language comparison was used as the criteria for conformance following the methodology of [5]. Application of the sweep-line state space method on IOTP was investigated in [25] exploiting that there is an inherent progression from the start of an IOTP transaction to termination of the transaction.

The Wireless Application Protocol (WAP) has been considered in [29, 30]. WAP is designed to provide Internet services to a wide range of hand-held devices. The work of [29, 30] concentrates on the Wireless Transaction Protocol (WTP) which is an important element of the WAP architecture and protocol suite. The work in [29] present a formal modelling of the WTP service and a formal modelling of the WTP protocol. Checking the conformance of the WTP protocol against the WTP service was done using finite-state language comparison. This approach succeeded in detecting several inconsistencies between the protocol and the service which was provided as input to the WAP forum responsible for the development of WAP. The sweep-line method was used in [30] to alleviate the state explosion problem and allow for the verification of larger configurations of WTP. The application of the sweep-line method allowed configurations with parameter settings of retransmission counters corresponding to the recommended setting for GSM and IP network to be verified.

The Session Initiation Protocol (SIP) is a widely used protocol for establishment of Internet multimedia session, and has been subject to formal modelling and validation in [16, 54]. The INVITE transactions has been formally analysed using state space exploration in [16, 54] leading to identification of undesired terminating states of the protocol when operating over an unreliable communication medium. Security aspects of SIP has been investigated in [55]. The work of [27] focus on the formal modelling of a SIP-based protocol for multi-channel service oriented architectures. A formalisation of SIP with the purpose of providing a framework model for presence architectures in mobile computing is presented in [26]. Another multimedia control protocol, the Capability Exchange Signalling (CES) protocol, has been formally modelled using CPNs and verified using state space exploration in [56]. The work on the CES protocol led to the identification of protocol errors in presence of sequence number wrap. Suggested changes were incorporated in a revised CPN model, and it was formally verified showing that the discovered errors have been eliminated.

The NEO protocol which is part of the distributed transactional object database management system NEOPPOD was investigated using high-level Petri Nets in [11]. The Coloane environment was used for the construction of the models, and verification was performed using the CPN-AMI and Helena tools. The NEO protocol is used to coordinate data storage and retrieval in a decentralised and distributed system where data can be stored on an number of data nodes and data is accessed through the primary master node. The focus of [11] was on the protocol used for the election of the primary master node. The model of the election part of the NEO protocol consisted of eighteen modules. Since there existed no specification document for the protocol, the Petri net model was reverse-engineered from a prototype implementation. The validation process which relied on the use of state spaces discovered two flaws in the implementation of the protocol. These were subsequently provided to the software engineers responsible for the implementation of the protocol component.

The Resource Reservation Protocol (RSVP) was formally modelled and verified in [80, 81]. The modelling and verification concentrates on verifying the absence of deadlocks and livelocks in relation to the setup, maintenance and path release procedures of RSVP. In addition, a number of RSVP specific behavioural properties were investigated which considered in detail the internal state of the sender, router, and receiver protocol entities of the protocol. The main contribution of the work was the development of a formal specification of the RSVP path procedures. Another example on the modelling of routing protocols can be found in [53] which uses Mobile Petri Nets to construct a formal model of the Mobile IP protocol. Mobile IP allows transport layer connections to preserved when mobile nodes changes their point of attachment to the Internet. CPNs have also recently been used for the verification of security protocols. Privacy enhancing protocols were considered in [73], and [28] address the modelling and validation of PANA Authentication and Authorisation Protocol.

6.3 Deriving Protocol Implementation Software

While CPNs have been successfully applied to modelling and validating protocol designs, there has been relatively few attempts at using the constructed CPN models in an automated or semi-automated manner as a basis for the actual implementation of protocols. Some simulation-based approaches were used in [63] and [48] for generating server-side implementations. Here, the simulation code for the CPN model generated by CPN Tools was extracted, and after undergoing automatic modifications (e.g., linking the code to external libraries), the generated simulation code is used as the system implementation. A limitation of this approach is that the execution speed is affected because each step in the execution of the program involves the computation and execution of enabled transitions (as done by a CPN simulator) in order to determine the next state. Secondly, the approach ties the target platform to that of the CPN Tools simulator which may make the approach impractical for many application domains due to resource consumption of the CPN simulator. The SML/NJ compiler used for the simulator in CPN Tools has a large memory footprint making it ill-suited, e.g., for the domain of embedded systems. Some initial work on a translation-based approach can be found in [52]. Here a restricted form of CPNs was used for obtaining an Erlang implementation of the DYMO routing protocol. The approach in [52] relies on the use of Process-Partitioned CPNs which enforces a detailed modelling of the protocol design which is very close to an implementation level model.

Acknowledgements. The authors acknowledge the contribution of Kristian L. Espensen and Mads. K. Kjeldsen in the project on the DYMO protocol, Paul Fleischer for his contribution in the GAN project, Michael Westergaard and Peder Christian Nørgaard for their contribution to the project on the RIP protocol, and Kurt Jensen for his contributions in the ERDP project.

References

1. 3GPP. Digital Cellular Telecommunications System (Phase 2+); Generic Access to the A/Gb Interface; Stage 2. 3GPP TS 43.318 version 6.9.0 Release 6, March 2007.
2. 3GPP. Website of 3GPP. `http://www.3gpp.org`, May 2007.
3. C. Baier and J-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
4. J. Billington, G. Gallasch, L.M. Kristensen, and T. Mailund. Exploiting Equivalence Reduction and the Sweep-Line Method for Detecting Terminal States. *IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans*, 34(1):23–38, 2004.
5. J. Billington, G. E. Gallasch, and B. Han. A Coloured Petri Net Approach to Protocol Verification. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 210–290. Springer, 2004.
6. J. Billington and B. Han. Modelling and Analysing the Functional Behaviour of TCPs Connection Management Procedures. *International Journal on Software Tools for Technology Transfer*, 9(3-4):269–304, 2007.
7. J. Billington and S. Vanit-Anunchai. Coloured Petri Net Modelling of an Evolving Internet Protocol Standard: The Datagram Congestion Control Protocol. *Fundamenta Informaticae*, 88(3):357–385, 2008.
8. J. Billington and C. Yuan. On modelling and analysing the dynamic manet on-demand (dymo) routing protocol. In *Transactions on Petri Nets and Other Models of Concurrency*, volume 5800 of *LNCS*, pages 98–126, 2009.
9. I.D. Chakeres and C.E. Perkins. Dynamic MANET On-demand (DYMO) Routing. `http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-10.txt`, July 2007. Internet-Draft. Work in Progress.
10. I.D. Chakeres and C.E. Perkins. Dynamic MANET On-demand (DYMO) Routing. `http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-11.txt`, November 2007. Internet-Draft. Work in Progress.
11. C. Choppy, A. Dedova, S. Evangelista, S. Hong, K. Klai, and L. Petrucci. The NEO Protocol for Large-Scale Distributed Database Systems: Modelling and Initial Verification. In *Proc. of ICATPN'10*, volume 6128 of *LNCS*, pages 145–164. Springer, 2010.
12. S. Christensen, J.B. Jørgensen, and L.M. Kristensen. Design/CPN – A Computer Tool for Coloured Petri Nets. In *Proc. of TACAS'97*, volume 1217 of *LNCS*, pages 209–223. Springer, 1997.

13. S. Christensen, L.M. Kristensen, and T. Mailund. A Sweep-Line Method for State Space Exploration. In *Proc. of TACAS'01*, volume 2031 of *LNCS*, pages 450–464. Springer, 2001.
14. E.M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting Symmetries in Temporal Logic Model Checking. *Formal Methods in System Design*, 9:77–104, 1996.
15. S. Deering and R. Hinden. Internet Protocol, Version 6 (IPV6) Specification, December 1998. RFC 2460.
16. L.G. Ding and L. Liu. Modelling and Analysis of the INVITE Transaction of the Session Initiation Protocol Using Coloured Petri Nets. In *Proc. of ICATPN'08*, volume 5062 of *LNCS*, pages 132–151. Springer, 2008.
17. E.A. Emerson and A.P. Sistla. Symmetry and Model Checking. *Formal Methods in System Design*, 9:105 – 131, 1996.
18. K.L. Espensen, M.K. Kjeldsen, and L.M. Kristensen. Modelling and Initial Validation of the DYMO Routing Protocol for Mobile Ad-Hoc Networks. In *Proc. of ICATPN'08*, volume 5062 of *LNCS*, pages 152–170. Springer, 2008.
19. S. Evangelista, M.Westergaard, and L.M. Kristensen. The ComBack Method Revisited: Caching Strategies and Extension with Delayed Duplicate Detection. *Transactions on Petri Nets and Other Models of Concurrency*, 3:189–215, 2009.
20. Examples of Industrial Use of CPNs.
`http://cs.au.dk/cpnets/industrial-use/`.
21. P. Fleischer and L.M. Kristensen. Modelling and Validation of Secure Connection Establishment in a Generic Access Network Scenario. *Fundamenta Informaticae*, 94(3-4):361–386, 2009.
22. G. E. Gallasch and J. Billington. Using parametric automata for the verification of the stop and wait class of protocols. In *Proc 3rd Int. Symposium on Automated Technology for Verification and Analysis*, volume 3707 of *LNCS*, pages 457–473. Springer, 2005.
23. G. E. Gallasch and J. Billington. A Parametric State Space for the Analysis of the Infinite Class of Stop-and-Wait Protocols. In *Proc. of SPIN Workshop on Model Checking of Software*, volume 3925 of *LNCS*, pages 201–218. Springer, 2006.
24. G. E. Gallasch, J. Billington, S. Vanit-Anunchai, and L.M. Kristensen. Checking Safety Properties On-the-fly with the Sweep-Line Method. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(3-4):371–392, 2007.
25. G.E. Gallasch, C. Ouyang, J. Billington, and L.M. Kristensen. Experimenting with Progress Mappings for the Sweep-Line Analysis of the Internet Open Trading Protocol. In *Proc. of 5th Workshop and Tutorial on Practical Use of Coloured Petri Nets and CPN Tools (CPN'04)*, pages 19–38, 2004.
26. V. Gehlot and A. Hayrapetyan. A Formalized and Validated Executable Model of the SIP-based Presence Protocol for Mobile Applications. In *Proceedings of the 45th Annual ACM Southeast Regional Conference*, pages 185–190. ACM, 2007.
27. Vijay Gehlot and Anush Hayrapetyan. A CPN Model of a SIP-Based Dynamic Discovery Protocol for Webservices in a Mobile Environment. In *Proc. of 7th Workshop and Tutorial onPractical Use of Coloured Petri Nets and the CPN Tools (CPN'06)*, 2006.
28. S. Gordon. Formal Analysis of PANA Authentication and Authorisation Protocol. In *Proc. of 9th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 277–284. IEEE Computer Society, 2008.
29. S. Gordon and J. Billington. Analysing the WAP Class 2 Wireless Transaction Protocol Using Coloured Petri Nets. In *Proc. of ICATPN'00*, volume 1825 of *LNCS*, pages 207–226. Springer, 2000.

30. S. Gordon, L.M. Kristensen, and J. Billington. Verification of a Revised WAP Wireless Transaction Protocol. In *Proc. of ICATPN'02*, volume 2360 of *LNCS*, pages 182–202. Springer, 2002.
31. P. Grimstrup. Interworking Description for IKEv2 Library. Ericsson Internal. Document No. 155 10-FCP 101 4328 Uen, September 2006.
32. G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.
33. The Internet Engineering Task Force (IETF). `http://www.ietf.org`.
34. C.N. Ip and D.L. Dill. Better Verification Through Symmetry. *Formal Methods in System Design*, 9:41 – 75, 1996.
35. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1:Basic Concepts*. Monographs in Theoretical Computer Science. Springer, 1992.
36. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2: Analysis Methods*. Monographs in Theoretical Computer Science. Springer, 1994.
37. K. Jensen. Condensed State Spaces for Symmetrical Coloured Petri Nets. *Formal Methods in System Design*, 9:7 – 40, 1996.
38. K. Jensen and L.M. Kristensen. *Coloured Petri Nets – Modelling and Validation of Concurrent Systems*. Springer, 2009.
39. K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(3-4):213–254, 2007.
40. J.B. Jørgensen and L.M. Kristensen. Computer Aided Verification of Lamport?s Fast Mutual Exclusion Algorithm Using Coloured Petri Nets and Occurrence Graphs with Symmetries. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):714–732, 1999.
41. J.B. Jørgensen and L.M. Kristensen. Verification of Coloured Petri Nets Using State Spaces with Equivalence Classes. In *Petri Net Approaches for Modelling and Validation*, volume 1 of *LINCOM Studies in Computer Science*, chapter 2, pages 17–34. Lincoln Europa, 2003.
42. C. Kaufman. Internet Key Exchange Protocol. RFC 4306, December 2005.
43. S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, December 2005.
44. L.M. Kristensen. A Perspective on Explicit State Space Exploration of Coloured Petri Nets: Past, Present, and Future. In *Proc. of ICATPN'10*, volume 6128 of *LNCS*, pages 39–42. Springer, 2010.
45. L.M. Kristensen and K. Jensen. Specification and Validation of an Edge Router Discovery Prot ocol for Mobile Ad-hoc Networks. In *Proc. of Integration of Software Specification Techniques for Applications in Engineering*, volume 3147 of *LNCS*, pages 248–269. Springer, 2004.
46. L.M. Kristensen and T. Mailund. A Generalised Sweep-Line Method for Safety Properties. In *Proc. of FME'02*, volume 2391 of *LNCS*, pages 549–567. Springer, 2002.
47. L.M. Kristensen and T. Mailund. Efficient Path Finding with the Sweep-Line Method using External Storage. In *Proc. of ICFEM'03*, volume 2885 of *LNCS*, pages 319–337. Springer, 2003.
48. L.M. Kristensen, P. Mechlenborg, L. Zhang, B. Mitchell, and G. E. Gallasch. Model-based Development of COAST. *STTT*, 10(1):5–14, 2007.

49. L.M. Kristensen, J.B. J ørgensen, and K. Jensen. Application of Coloured Petri Nets in System Development. In *Proc. of 4th Advanced Course on Petri Nets Lectures on Concurrency and Petri Nets - Advances in Petri Nets*, volume 3098 of *LNCS*, pages 626–685. Springer, 2004.
50. L.M. Kristensen and A. Valmari. Finding Stubborn Sets of Coloured Petri Nets Without Unfolding. In *Proc. of ICATPN'98*, volume 1420 of *LNCS*, pages 104–123. Springer, 1998.
51. L.M. Kristensen, M. Westergaard, and P.C. Nrgaard. Model-based Prototyping of an Interoperability Protocol for Mobile Ad-hoc Networks. In *Proc. of Fifth International Conference on Integrated Formal Methods (IFM05)*, volume 3771 of *LNCS*, pages 266–286. Springer, 2005.
52. L.M. Kristensen and Michael Westergaard. Automatic Structure-Based Code Generation from Coloured Petri Nets: A Proof of Concept. In *Proc. of FMICS'2010*, volume 6371 of *LNCS*, pages 215–230. Springer, 2010.
53. C. Lakos. Modelling Mobile IP with Mobile Petri Nets. *Transactions on Petri Nets and Other Models of Concurrency*, 5800:127–158, 2009.
54. L. Liu. Verification of the sip transaction using coloured petri nets. In Bernard Mans, editor, *Thirty-Second Australasian Computer Science Conference (ACSC 2009)*, volume 91 of *CRPIT*, pages 63–72, Wellington, New Zealand, 2009. ACS.
55. L. Liu. Security Analysis of Session Initiation Protocol - A methodology based on Coloured Petri Nets. In *Proc. of the 2010 International Cyber Resilience Conference*, 2010.
56. L. Liu and J. Billington. Verification of the Capability Exchange Signalling protocol. *STTT*, 9(3-4):305–326, 2007.
57. Ming T. Liu. Protocol Engineering. *Advances in Computers*, 29:79–195, 1989.
58. L. Lorentsen and L.M. Kristensen. Modelling and Analysis of a Danfoss Flowmeter System using Coloured Petri Nets. In *Proc. of ICATPN'00*, volume 1825 of *LNCS*, pages 346–366. Springer, 2000.
59. L. Lorentsen and L.M. Kristensen. Exploiting Stabilizers and Parallelism in State Space Generation with the Symmetry Method. In *Proc. of ICACSD'01*, pages 211–220. IEEE Computer Society, 2001.
60. IETF Mobile Ad-hoc Networks Discussion Archive.
`http://www1.ietf.org/mail-archive/web/manet/current/index.html`.
61. T. Mailund. Analysing Infinite-State Systems by Combining Equivalence Reduction and the Sweep-Line Method. In *Proc. of ICATPN'02*, volume 2360 of *LNCS*, pages 314–333. Springer, 2002.
62. G. Malkin. RIP Version 2. RFC 4822, February 2007.
63. K.H. Mortensen. Automatic Code Generation Method Based on Coloured Petri Net Models Applied on an Access Control System. In *Proc. of ICATPN'00*, volume 1825 of *LNCS*, pages 367–386. Springer, 2000.
64. T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6), December 1998. RFC 2461.
65. C. Ouyang and J. Billington. On Verifying the Internet Open Trading Protocol. In *Proc. of 4th International EC-Web Conference*, volume 2738 of *LNCS*, pages 292–302. Springer, 2003.
66. C. Ouyang and J. Billington. Formal Analysis of the Internet Open Trading Protocol. In *Proc. of Applying Formal Methods: Testing, Performance and M/ECommerce, FORTE 2004 Workshops*, volume 3236 of *LNCS*, pages 1–15. Springer, 2004.

67. C. Ouyang, L.M. Kristensen, and J. Billington. A Formal and Executable Specification of the Internet Open Trading Protocol. In *Proc. of Third International Conference on E-Commerce and Web Technologies*, volume 2455 of *LNCS*, pages 377–387. Springer, 2002.
68. C. Ouyang, L.M. Kristensen, and J. Billington. A Formal Service Specification for the Internet Open Trading Protocol. In *Proc. of ICATPN'02*, volume 2360 of *LNCS*, pages 352–373. Springer, 2002.
69. C.A. Petri. *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.
70. A.V. Ratzer, L. Wells, H.M. Lassen, M. Laursen, J.F. Qvortrup, M.S. Stissing, M. Westergaard, S. Christensen, and K. Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In *Proc. of ICATPN'03*, volume 2679 of *LNCS*, pages 450–462. Springer, 2003. `http://www.cpntools.org`.
71. W. Reisig. *Petri Nets - An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985.
72. U. Stern and D.L. Dill. Improved Probabilistic Verification by Hash Compaction. In *Proc. of Correct Hardware Design and Verification Methods*, volume 987 of *LNCS*, pages 206–224. Springer, 1995.
73. S. Suriadi, C. Ouyang, J. Smith, and E. Foo. Modeling and Verification of Privacy Enhancing Protocols. In *Proc. of ICFEM'09*, volume 5885 of *LNCS*, pages 127–146. Springer, 2009.
74. J.D. Ullman. *Elements of ML Programming.* Prentice-Hall, 1998.
75. A. Valmari. A Stubborn Attack on State Explosion. In *Proc. of Computer-Aided Verification (CAV'90)*, volume 531 of *LNCS*, pages 156–165. Springer, 1990.
76. A. Valmari. Stubborn Sets of Coloured Petri Nets. In *Proc. of ICATPN'91*, pages 102–121, 1991.
77. A. Valmari. The State Explosion Problem. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 429–528. Springer, 1998.
78. S. Vanit-Anunchai, J. Billington, and G. E. Gallasch. Analysis of the Datagram Congestion Control Protocols Connection Management Procedures using the Sweep-line Method. *International Journal on Software Tools for Technology Transfer*, 10(1):29–56, 2008.
79. Somsak Vanit-Anunchai. Towards Formal Modelling and Analysis of SCTP Connection Management. In *Proc. of CPN'09*, pages 163–182, 2008.
80. M.E. Villapol and J. Billington. A Coloured Petri Net Approach to Formalising and Analysing the Resource Reservation Protocol. *CLEI Electron. J.*, 6(1), 2003.
81. M.E. Villapol and J. Billington. Analysing Properties of the Resource Reservation Protocol. In *Proc. of ICATPN'03*, volume 2679 of *LNCS*, pages 377–396. Springer, 2003.
82. P. Vixie. Dynamic Updates in the Domain Name System. RFC 2136, April 1997.
83. M. Westergaard, S. Evangelista, and L.M. Kristensen. ASAP: An Extensible Platform for State Space Analysis. In *Proc. of ICATPN'09*, volume 5606 of *LNCS*, pages 303–312. Springer, 2009. `http://www.daimi.au.dk/~ascoveco/download.html`.
84. M. Westergaard, L.M. Kristensen, G.S. Brodal, and L.A. Arge. The ComBack Method – Extending Hash Compaction with Backtracking. In *Proc. of ICATPN'07*, volume 4546 of *LNCS*, pages 445–464. Springer, 2007.
85. M. Westergaard and K. B. Lassen. The BRITNeY Suite Animation Tool. In *Proc. of ICATPN'06*, volume 4024 of *LNCS*, pages 431–440. Springer, 2006.
86. P. Wolper and D. Leroy. Reliable Hashing without Collision Detection. In *Proc. of CAV'93*, volume 697 of *LNCS*, pages 59–70. Springer, 1993.