Specification and Validation of an Edge Router Discovery Protocol for Mobile Ad Hoc Networks

Lars Michael Kristensen[⋆] and Kurt Jensen

Department of Computer Science, University of Aarhus
IT-parken, Aabogade 34, DK-8200 Aarhus N, DENMARK,
{lmkristensen,kjensen}@daimi.au.dk

Abstract. We present an industrial project at Ericsson Telebit A/S where Coloured Petri Nets (CP-nets or CPNs) have been used for the design and specification of an edge router discovery protocol for mobile ad-hoc networks. The Edge Router Discovery Protocol (ERDP) supports an edge router in a stationary core network in assigning network address prefixes to gateways in mobile ad-hoc networks. This paper focuses on how CP-nets and the CPN computer tools have been applied in the development of ERDP. A CPN model has been constructed that constitutes a formal executable specification of ERDP. Simulation and message sequence charts were used for initial investigations of the protocol's behaviour. Then state space analysis was applied to conduct a formal verification of the key properties of ERDP. Both the modelling, simulation, and subsequent state space analysis helped in identifying several omissions and errors in the design, demonstrating the benefits of using formal modelling and analysis in a protocol design process.

1 Introduction

The specification and development of communication protocols is a complex task. One of the reasons is that protocols consist of a number of independent concurrent protocol entities that may proceed in many different ways depending on when, e.g., packets are lost, timers expire, and processes are scheduled. The complex behaviour makes the design of correct protocols a challenging task.

The specification of communication protocols 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) [17]. Natural language specifications of protocols have several weaknesses. Firstly, they are inherently ambiguous making it difficult to achieve inter-operability between independent implementations. Secondly, they are often incomplete in the sense that the behaviour of the protocol is not completely described for all cases. This has motivated the use of formal description techniques [3] for the specification and validation of protocols.

An advantage of many formal description techniques is that they are based on the construction of executable models that make it possible to observe and

[⋆] Supported by the Danish Natural Science Research Council.

experiment with the behaviour of the protocol prior to implementation using, e.g., simulation. This typically leads to complete specifications since the model will not be fully operational until all parts of the protocol have been specified. Another advantage of formal modelling languages is that they support abstractions, making it possible to specify the operation of the protocol without being concerned with irrelevant implementation details. Furthermore, the use of formal description techniques results in models that are amenable to verification using, e.g., state space methods where the basic idea is to compute all reachable states and state changes of the protocol and represent these as a directed graph. State space methods make it possible to automatically (i.e., algorithmically) check whether a protocol has the desired set of properties. 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. Other examples of verification techniques for Petri nets [25, 10] includes structural and invariant methods [8, 26], and partial-order and unfolding methods [9, 13].

We present a case study from a joint research project [20] between the Coloured Petri Nets Group [6] at University of Aarhus and Ericsson Telebit A/S [12]. The research project applies formal description techniques in the form of Coloured Petri Nets (CP-nets or CPNs) [18, 21] and the supporting CPN computer tools [5, 11] in the development of Internet Protocol Version 6 (IPv6) [15] based protocols for ad-hoc networking [23]. An ad-hoc network is 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 in the ad-hoc network operate in a fully self-configuring and distributed manner, without any preexisting communication infrastructure such as base stations and routers. Network layer and routing protocols for ad-hoc networking are under development by the IETF Mobile Ad-hoc Networks working group [14].

The CPN modelling language combines Petri nets and programming languages. Petri nets [25, 10] provide the foundation of the graphical notation and the semantical foundation for modelling concurrency, synchronisation, and communication in systems. The functional programming language Standard ML [27] provides the primitives for compactly modelling the sequential aspects of systems (such as data manipulation) and for creating compact and parameterisable models. State space methods are the main verification techniques for CP-nets. The CPN modelling language is supported by two computer tools: CPN Tools [5] and Design/CPN [11].

This paper shows how CP-nets and the CPN computer tools have been used and integrated in the development of the Edge Router Discovery Protocol (ERDP). ERDP is a protocol for connecting gateways in mobile ad-hoc networks to edge routers in stationary core networks. ERDP allows gateway nodes to discover and create an attachment to the core network, and it allows edge routers to configure gateways with a globally routeable IPv6 address prefix. This address prefix can, in turn, be used by the gateway to configure global IPv6 unicast addresses for the mobile nodes in the ad-hoc network. CP-nets have previously been applied in a number of industrial projects for the specification and analysis of protocols and systems [16].

The paper is organised as follows. Section 2 gives a brief introduction to ERDP explaining the basic operation of the protocol. Section 3 presents selected parts of the CPN model specifying ERDP, and Sect. 4 shows how the protocol was analysed and verified using state spaces. Finally, in Sect. 5 we sum up the conclusions. The reader is assumed to be familiar with the basic ideas of Petri nets, high-level Petri nets, and state space methods. The paper [21] gives an introduction to CP-nets sufficient to understand the results presented in this paper. The reader is referred to [18] for a complete introduction to CP-nets, their analysis methods, and applications.

2 The Edge Router Discovery Protocol

This section gives a brief introduction to the Edge Router Discovery Protocol (ERDP). The aim is not to give a complete description of ERDP, but to provide sufficient information to understand the CPN model and the state space analysis results presented in the later sections.

Figure 1 shows the IPv6-based network architecture considered in the project. The network architecture consists of an IPv6 stationary core network connecting a number of mobile ad-hoc networks on the edge of the core network. The project focuses on IP routing in this hybrid network architecture. A number of *edge routers* reside on the edge of the core network, and an ad-hoc network may contain one or more nodes capable of acting as *gateways* for communication with nodes outside the ad-hoc network. The edge routers and the gateways handle the connection between the core network and the ad-hoc networks, and an edge router may serve multiple ad-hoc networks. The core network is a classical wired IP network with stationary nodes, whereas wireless communication is used for communication between the mobile nodes in the ad-hoc networks. The edge routers and the gateways are also connected via wireless links that may be different from the wireless link technology used internally in the ad-hoc networks.

The network architecture also encompasses mobility. The nodes in the individual ad-hoc networks may move within the ad-hoc network or between the ad-hoc networks. It is also possible for an entire ad-hoc network, including its

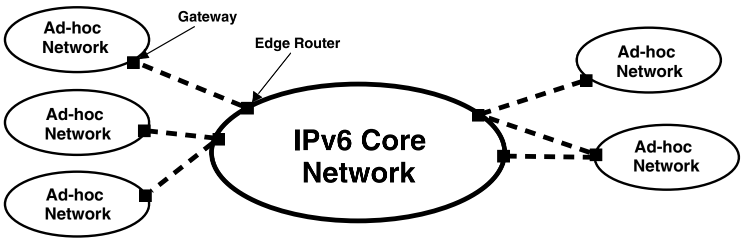


Fig. 1. IPv6-based Networking Architecture.

gateways to move from one edge router to another edge router, and possibly be within reach of several edge routers simultaneously.

ERDP is used between the gateways in the ad-hoc networks and the edge routers in the core network. ERDP supports gateways in discovering edge routers and supports edge routers in configuring gateways with a globally routeable IPv6 address prefix. This address prefix can then be used to configure global IPv6 unicast addresses for mobile nodes in the ad-hoc networks. This has the effect that all nodes in a given ad-hoc network will have addresses based on the same IPv6 address prefix, and entries in the routing tables in the core network can therefore be based on address prefixes rather than having to contain the individual addresses of the ad-hoc network nodes. This ensures that the network architecture scales with respect to the number of entries in the routing tables. When an ad-hoc network moves from one edge router to another edge router, it must be configured with a new topologically correct IPv6 address prefix for the same scalability reasons. Gateway discovery and selection by nodes in the ad-hoc networks are handled by protocols executed internally in the ad-hoc networks and is outside the scope of ERDP.

ERDP is based on extending the Neighbour Discovery Protocol (NDP) [22] which is part of the IPv6 protocol suite. The intended use of NDP is for hosts to discover neighbouring nodes and routers on a local area network, and conduct address configuration. In the IPv6-based networking architecture in Fig. 1, gateways acts as hosts and edge routers acts as routers from the point of view of NDP. Standard NDP can however not be used for configuration of gateways with address prefixes in the above network architecture since it must be ensured that a given address prefix is only assigned to a single gateway. This is required to ensure that routing of packets from edge routers to gateways is well-defined in case an ad-hoc network splits into two ad-hoc networks each with their own gateway.

Figure 2 shows the basic way that an edge router configures a gateway with an address prefix using ERDP. The message sequence chart was generated automatically from the CPN model to be presented in Sect. 3. The column labelled GWBuffer represents packet buffers between the gateway protocol entity and the underlying protocol layers. Similarly, the ERBuffer column represents packet buffers in the edge router.

An edge router periodically multi-casts unsolicited router advertisements (RAs) to announce its presence to any gateways that may be within reach of the edge router. When an unsolicited RA is received by a gateway, it will reply with its list of currently assigned address prefixes in a unicast router solicitation (RS). In this example, the gateway has no current prefixes and hence it sends an RS with no prefixes (indicated by the empty list [ ]). When the edge router receives the RS it will consult its lists of available prefixes and in this case select a new address prefix (P1) to be assigned to the gateway. This newly assigned prefix will then be sent back to the gateway in a unicast RA. When the unicast RA containing the prefix is received by the gateway, it will update its lists of currently assigned prefixes to contain the new prefix P1. Prefixes assigned to

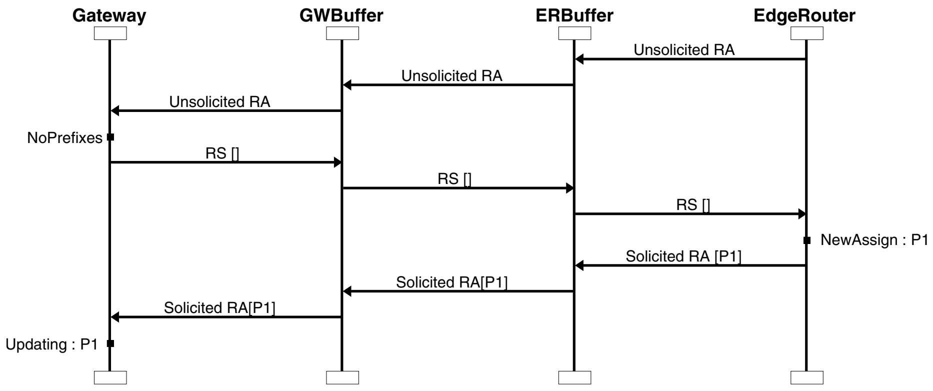


Fig. 2. Message sequence chart for basic prefix configuration with ERDP.

gateways have a limited lifetime, and hence either will expire or will have to be refreshed by the edge router.

CP-nets were integrated in the design and specification of ERDP by developing a CPN model of ERDP together with a conventional natural language specification. In the following sections we will refer to the natural language specification of ERDP as the *ERDP specification.*

3 CPN Modelling

Figure 3 shows the *hierarchy page* of the CPN model providing an overview of the *pages* (modules) constituting the CPN model. Each node in Figure 3 represents a page in the CPN model, and is labelled with a page name and a page number. As an example, the page node at the top left is named ERDP and has page number 1. Page ERDP is the most abstract page in the CPN model. An arc between two nodes indicates that the destination page is a subpage (submodule) of the source page. The arc label(s) specifies the name of the *substitution transition*(s) representing the corresponding subpage at the source page.

The CPN model consists of three main parts. Page Gateway and its four subpages model the operation of the gateway. Page EdgeRouter and its five subpages model the operation of the edge router. Page GW_ER_Link models the wireless communication link between the gateway and the edge router. We consider a system consisting of one gateway and one edge router as this is sufficient for specifying the protocol itself, and for investigating its basic operation. Validation of ERDP in an environment with multiple gateways and edge routers is beyond the scope of this paper.

Figure 4 depicts page EDRP. It corresponds to the ERDP page node in Fig. 3. The substitution transition Gateway represents the gateway, and the substitution transition Edge Router represents the edge router. The communication link

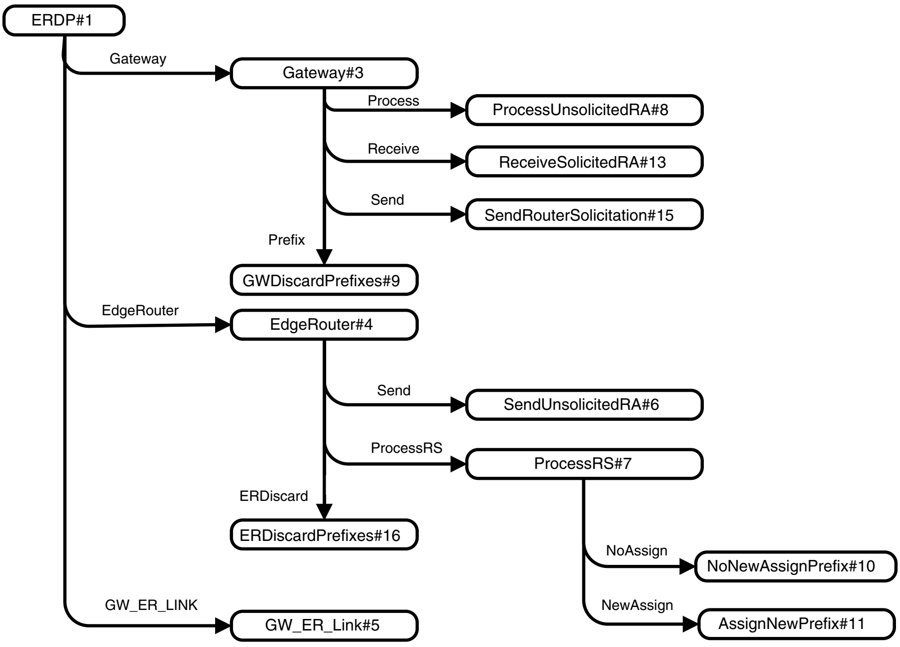


Fig. 3. Hierarchy page - overview of CPN model.

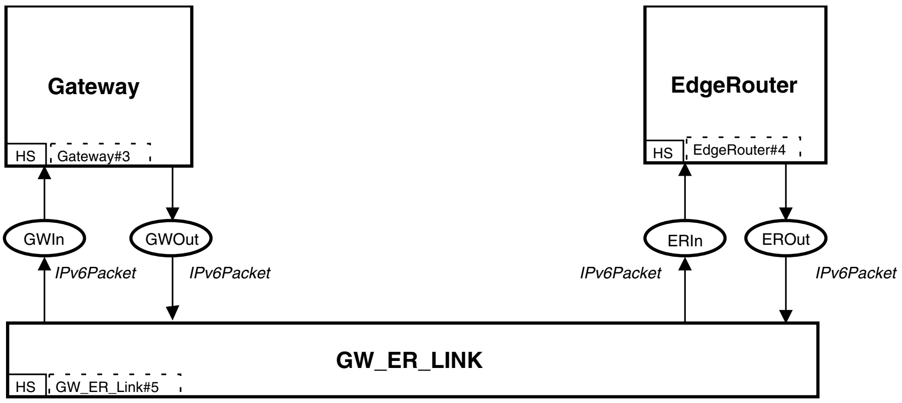


Fig. 4. The ERDP page - top level page in the CPN model.

between the edge router and the gateway is represented by the substitution transition GW_ER_LINK. The four places GWIn, GWOut, ERIn, and EROut model packet buffers between the link layer and the gateway and edge router. Both the edge router (ER) and the gateway (GW) have an incoming and an outgoing packet buffer.

3.1 Modelling Packets

All four places in Fig. 4 have the colour set (type) IPv6Packet used to model the IPv6 packets exchanged between edge routers and gateways. Since ERDP is based on the IPv6 Neighbour Discovery Protocol (NDP) [22], the packets are carried as Internet Control Message Protocol (ICMP) [4] packets. The definitions of colour sets for NDP, ICMP, and IPv6 packets are given in Fig. 5 and have been derived from RFC 2460 [7] specifying IPv6 and RFC 2461 [22] defining NDP. IPv6 addresses and address prefixes are modelled as strings. This makes it possible to use both mnemonic names and standard hexadecimal notation for IPv6 addresses in the CPN model. Protocol fields that do not affect the operation of ERDP have been defined using the colour set NOTMOD containing the single dummy value notmod. These fields could alternatively have been omitted, but for the later implementation of ERDP it was considered important that the tokens in the CPN model have the same set of fields as the packets in the implementation. The colour sets UInt8, Bit4, and Bit8 (not shown) used for bit fields in the packets are all defined as integers, as we are not concerned with the specific layout of packets and the domain of the bit fields, but only their semantics.

3.2 Modelling the Edge Router

Figure 6 shows the page EdgeRouter modelling the edge router. The *port* places ERIn and EROut are assigned to the similarly named *socket* places on page ERDP (see Fig. 4). The place Config models the configuration information associated with the edge router, and place PrefixPool 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 7 shows the declarations of the colour sets for the three places in Fig. 6. 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 place PrefixPool 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 place PrefixPool which is by default set to 1.

The substitution transition SendUnsolicitedRA (in Fig. 6) corresponds to the multi-cast of periodic unsolicited Router Advertisements (RAs) by the edge router. The substitution transition ProcessRS models the reception of unicast Router Solicitations (RSs) from gateways, and the sending of a unicast RA in response. The substitution transition ERDiscardPrefixes models expiration of prefixes on the edge router side.

The marking shown in Fig. 6 has a single token on each of the three places used to model the internal state of the edge router protocol entity. This is shown by the small circles positioned on top of these places and containing the integer

```
color IPv6Addr = string; (* IPv6 addresses *)

(* --- Router Soliciations --- *)
color RSOption = union RS_SrcLinkAddr       : NDLinkAddrOption +
                       RS_PrefixInformation : NDPrefixInformationOption;
color RSOptions = list RSOption;

color RouterSolicitation = record Options  : RSOptions *
                                  NU       : NOTMOD;

(* --- Router Advertisements --- *)
color RAOption = union RA_SrcLinkAddr        : NDLinkAddrOption +
                       RA_MTU                : NDMTUOption      +
                       RA_PrefixInformation  : NDPrefixInformationOption;
color RAOptions = list RAOption;

color RouterAdvertisement = record CurHopLimit    : UInt8  *
                                   M              : Bit    *
                                   O              : Bit    *
                                   RouterLifetime : UInt16 *
                                   ReachableTime  : UInt32 *
                                   RetransTimer   : UInt32 *
                                   Options        : RAOptions;

(* --- ICMP messages --- *)
color ICMPBody = union RS : RouterSolicitation     +
                       RA : RouterAdvertisement;
color ICMPMessage = record Type : UInt8 *
                           Code : UInt8 *
                           Message : ICMPBody;

(* --- IPv6 packets --- *)
color IPv6Payload = union ICMP : ICMPMessage;
color IPv6Header = record Version             : Bit4     *
                          TrafficClass        : NOTMOD   *
                          Flowlabel           : NOTMOD   *
                          PayloadLenght       : NOTMOD   *
                          NextHeader          : Bit8     *
                          HopLimit            : Bit8     *
                          SourceAddress       : IPv6Addr *
color IPv6Packet = record header     : IPv6Header *
                          extheaders : NOTMOD *
                          payload    : IPv6Payload;
```

Fig. 5. Declarations for IPv6 and ICMP packets.

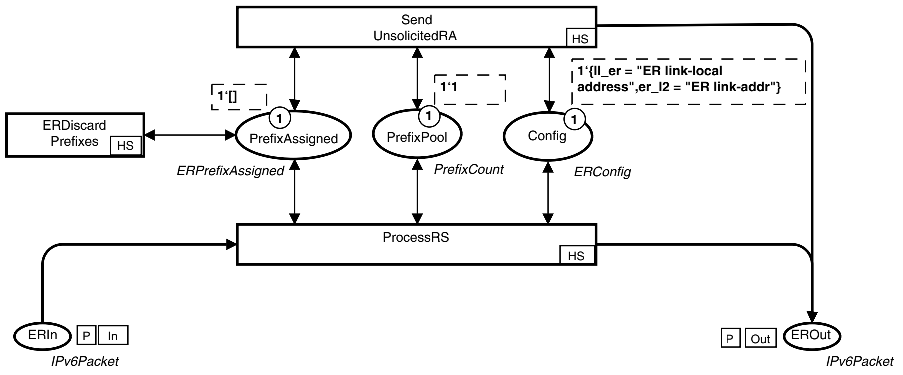


Fig. 6. The EdgeRouter page.

```
color LinkAddr = string;
color PrefixCount = int;

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

color ERPrefixEntry = product  IPv6Addr * IPv6Prefix;
color ERPrefixAssigned = list ERPrefixEntry;
```

Fig. 7. Declarations for modelling edge routers.

1. The dashed box positioned next to each of the small circles gives information about the colour of the token. In the marking shown, the token on place PrefixAssigned with the colour [] (empty list) corresponds to the edge router not having assigned any prefixes to the gateway. The token on place PrefixPool with colour 1 indicates that the edge router has a single prefix available for distribution. Finally, the colour of the token on place Config specifies the link-local and link address of the edge router. In this case the edge router has the symbolic link-local address of ER link-local address, and the symbolic link-address of ER link-addr.

Figure 8 depicts page SendUnsolicitedRA which is the subpage of the substitution transition SendUnsolicitedRA in Fig. 6. The transition SendUnsolicitedRA models the sending of the periodic unsolicited router advertisements. The variable erconfig is of type ERConfig (see Fig. 7) and the variable prefixleft is of type PrefixCount (see Fig. 7). The transition SendUnsolicitedRA is only enabled 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.

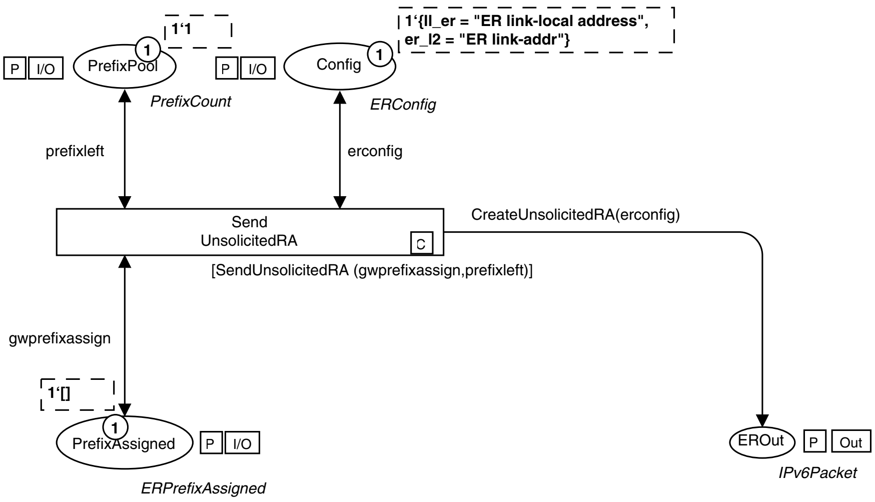


Fig. 8. Page SendUnsolicitedRA - initial state.

Figure 9 depicts the marking of page SendUnsolicitedRA after the occurrence of the transition SendUnsolicitedRA in the marking shown in Fig. 8. An unsolicited router advertisement has been put in the outgoing buffer of the edge router. It can be seen that the unsolicited router advertisement is sent to the all-nodes multi-cast address, and that it carries the link-local and link-layer address of the edge router as part of the options in the router advertisement.

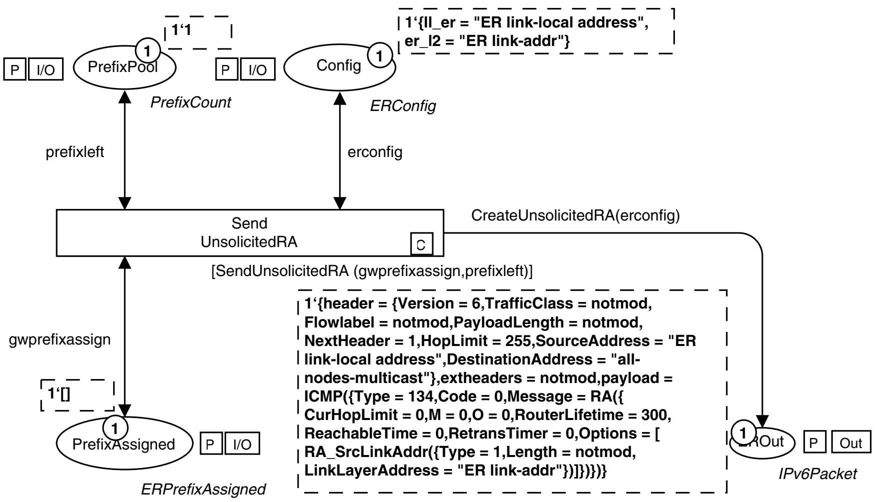

1'{ll_er = "ER link-local address",
er_l2 = "ER link-addr"}
1'1
PrefixPool
P I/O
PrefixCount
Config
ERConfig
prefixleft
erconfig
Send UnsolicitedRA
C
CreateUnsolicitedRA(erconfig)
[SendUnsolicitedRA (gwprefixassign,prefixleft)]
1'{header = {Version = 6,TrafficClass = notmod,
Flowlabel = notmod,PayloadLength = notmod,
NextHeader = 1,HopLimit = 255,SourceAddress = "ER
link-local address",DestinationAddress = "all-
nodes-multicast"},extheaders = notmod,payload =
ICMP({Type = 134,Code = 0,Message = RA({
CurHopLimit = 0,M = 0,O = 0,RouterLifetime = 300,
ReachableTime = 0,RetransTimer = 0,Options = [
RA_SrcLinkAddr({Type = 1,Length = notmod,
LinkLayerAddress = "ER link-addr"})]})})}
gwprefixassign
1'[]
PrefixAssigned
ERPrefixAssigned
EROut
P Out
IPv6Packet

Fig. 9. Page SendUnsolicitedRA - after occurrence of SendunsolicitedRA.

3.3 Modelling the Wireless Link

Figure 10 shows the part of page GW_ER_Link modelling 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 places GWIn and EROut are linked to the similarly named socket places in Fig. 4. 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 from the edge router to the gateway corresponds to moving the token modelling the packet from place EROut to GWIn. If the packet is lost, it will only be removed from place EROut.

Wireless links in general have lower bandwidth and higher error-rate than wired links. These characteristics have been abstracted away in the CPN model since our aim is not to reason about the performance of ERDP but rather its logical correctness. Duplication and reordering of messages is not possible on typical 1-hop wireless links since detection of duplicates and preservation of order will be handled by the data-link layer. The modelling of the wireless link does allow overtaking of packets, but this overtaking will be eliminated in the analysis phase described in Sect. 4 where we impose bounds on the capacity of the input and output packet buffers.

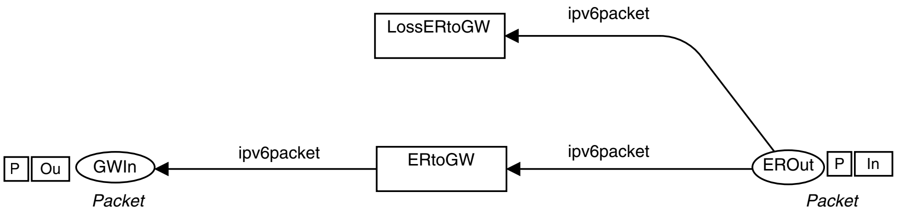


Fig. 10. Part of the GW_ER_Link page.

3.4 Summary of the Modelling Process

The CPN model presented above was developed as an integrated part of the development of ERDP. Creation of the CPN model was done by researchers from the Coloured Petri Nets group whereas the development of the ERDP specification was done by protocol developers at Ericsson Telebit A/S. Altogether 70 man-hours were spent on CPN modelling. The protocol developers at Ericsson Telebit A/S had been given a 6 hour course on the CPN modelling language. This course enabled them to read and interpret CPN models, allowing the CPN model to be used as a basis for discussions of the protocol design and its representation as a CPN model.

The development of ERDP started out with the creation of an initial ERDP (natural language) specification. Based on this specification, a first version of the CPN model was created. The act of creating this initial CPN model and

Table 1. Summary and categorisation of issues encountered in the modelling phase.

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

discussing it (in the following referred to as Review 1) lead to the identification of several issues related to the design and operation of ERDP. This included design errors, incompleteness and ambiguity in the specification, and ideas for simplifications and improvements of the protocol design. Based on the issues discovered in Review 1, the ERDP specification was revised and extended. A second review (Review 2) was then conducted by revising the initial CPN model according to the modified ERDP specification and then discussing the CPN model. Review 2 lead to further identification of issues which were eventually resolved and the ERDP specification was modified accordingly. The CPN model was then modified again to reflect the revised ERDP specification. At this stage, no further issues were discovered in the process of revising the CPN model.

Table 1 categorises and enumerates the issues encountered in each of the two reviews (Rev 1 and Rev 2). These issues were identified in the process of constructing the CPN model, single step execution of the CPN model, and discussions of the CPN model among the project group members. Altogether 24 issues were identified in the process of constructing and simulating the model.

Message Sequence Charts (MSCs) (such as the one shown in Fig. 2) integrated with simulation was used in both review steps to investigate the behaviour of ERDP in detail. The basic idea in this integration is to use MSCs to provide visual feedback from simulations of the CPN model. Visual feedback in the form of MSCs is supported by the MSC library [1] available for the CPN computer tools. Technically the integration of MSCs and simulation is achieved by the modeller attaching code segments to the transitions of the CPN model. The code segments invoke primitives in the MSC library. When a transition occurs in a simulation the associated code segment (if any) is executed causing the MSC to be accordingly updated. The visualisation at the level of the CPN model provides a state-based view whereas MSCs provides an event-based view that includes history. The two forms of feedback therefore complements each other.

4 State Space Analysis and Verification

State space analysis was pursued after the three iterations of modelling summarised at the end of the previous section. The purpose of the state space analysis was to conduct a more thorough investigation of the operation of ERDP, including verification of its key properties.

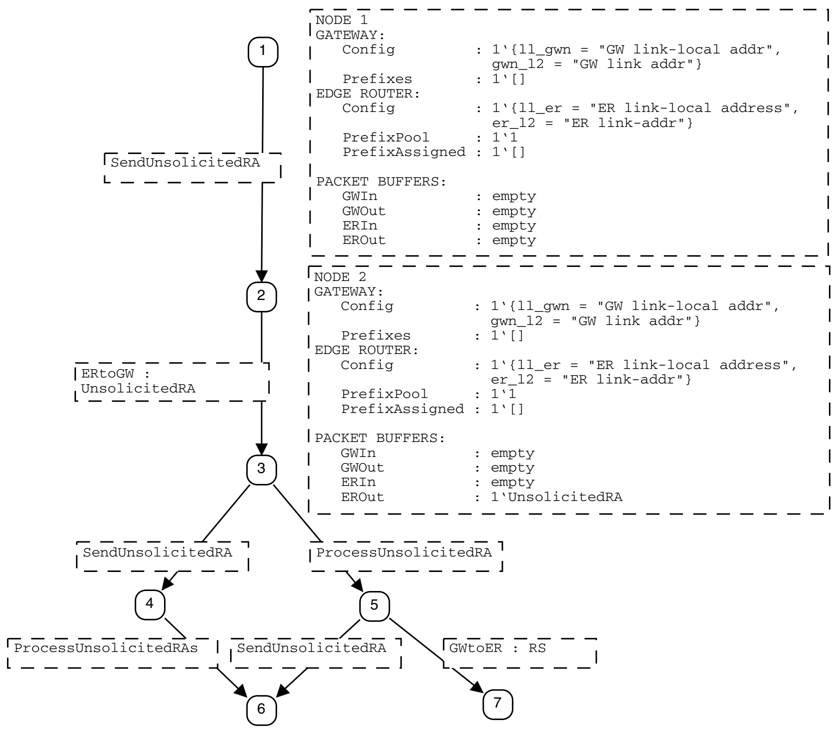
4.1 Analysis Preparations and Approach

The first step towards state space analysis of the CPN model was to obtain a finite state space. The CPN model presented in the previous section 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 bound of one token was imposed on each of the places GWIn, GWOut, ERIn, and EROut (see Fig. 4) modelling the packet buffers. This has the effect of also preventing reordering of the packets when transmitted across the wireless link. Furthermore, the number of packets simultaneously in the input and output buffers of the two protocol entities was limited to 2. Technically this was done by using *branching options* available in the CPN state space tool to not explore enabled transitions whose occurrence in a given marking would violate the above bounds.

Figure 11 shows the initial part of the state space in the case where we assume that packets cannot be lost when transmitted across the wireless link. Each of the nodes 1-7 represent a state of the CPN model, and each of the arcs represent an occurrence of an enabled transition. The labels on the arcs give the name of the transition occurring. Node 1 represents the initial state of the CPN model. The two dashed boxes associated with nodes 1 and 2 show the marking of each of the places in the CPN model in the corresponding state. It can be seen that initially (node 1) all packet buffers are empty, the edge router has one prefix to assign, and the gateway is not configured with any prefixes. Sending of an unsolicited RA by the edge router is the only possible event in the initial state. If this event occurs, the state corresponding to node 2 is entered. The state represented by node 2 is identical to the state represented by node 1, except that an unsolicited RA is now in the output buffer of the edge router. In state 2 the unsolicited RA can be sent to the gateway leading to state 3 in which either another unsolicited RA can be sent by the edge router or the gateway can process the unsolicited RA.

The state space analysis was based on first generating the state space for the considered configuration of the protocol. This was followed by generation of the *state space report* and the use of *query functions* to investigate the properties of the protocol. The state space report can be generated automatically by the CPN state space tool, and it contains information about a set of standard properties of the CPN model such as *integer bounds* (minimal/maximal number of tokens on places), *dead markings* (states without enabled transitions), and *home markings* (states that can always be reached). These properties are system independent in that the properties that can be investigated for any CPN model.

The query functions in the CPN state space tool support verification and analysis of system dependent properties. The query functions provide a set of primitives for traversing the state space in various ways, such as visiting all states. The analysis of ERDP relied mainly on the use of the *Strongly Connected Component* graph (SCC-graph) derived from the state space. The SCC-graph is generated by the CPN state space tool and is derived from the state space by considering states to be equivalent if they are mutually reachable. The SCC-



```
NODE 1
GATEWAY:
   Config          : 1`{ll_gwn = "GW link-local addr",
                       gwn_l2 = "GW link addr"}
   Prefixes        : 1`[]
EDGE ROUTER:
   Config          : 1`{ll_er = "ER link-local address",
                       er_l2 = "ER link-addr"}
   PrefixPool      : 1`1
   PrefixAssigned  : 1`[]

PACKET BUFFERS:
   GWIn            : empty
   GWOut           : empty
   ERIn            : empty
   EROut           : empty
```

```
NODE 2
GATEWAY:
   Config          : 1`{ll_gwn = "GW link-local addr",
                       gwn_l2 = "GW link addr"}
   Prefixes        : 1`[]
EDGE ROUTER:
   Config          : 1`{ll_er = "ER link-local address",
                       er_l2 = "ER link-addr"}
   PrefixPool      : 1`1
   PrefixAssigned  : 1`[]

PACKET BUFFERS:
   GWIn            : empty
   GWOut           : empty
   ERIn            : empty
   EROut           : 1`UnsolicitedRA
```

Fig. 11. Initial fragment of state space.

graph has a node for each such equivalence class containing the subgraph of the state space determined by the states in the equivalence class. A node in the SCC-graph is called a *Strongly Connected Component* (SCC). The SCC-graph is an acyclic graph, and two states s_1 and s_2 are in the same SCC S in the SCC-graph if and only if there is a path from s_1 to s_2 and a path from s_2 to s_1 in the state space. There is an arc in the SCC-graph between two SCCs S_1 and S_2 if there is a state s_1 in S_1 with an outgoing arc in the state space leading to a state s_2 contained in S_2. The *terminal SCCs* are the nodes in the SCC-graph without outgoing arcs. A SCC is *trivial* if it contains one state and no arcs. A number of query functions are available in the CPN state space tool for inspecting and traversing the SCC-graph.

The key property of ERDP is the proper configuration of the gateway with prefixes. By this we mean that for a given prefix and state where the gateway has not yet been configured with that prefix, the protocol must be able to configure the gateway with the prefix. Furthermore, when the gateway has been configured with the prefix, the edge router and the gateway should be properly synchronised, i.e., the assignment of the prefix must be recorded in the gateway protocol entity as well as in the edge router protocol entity. In the following we will refer to a state where the gateway is configured with a prefix and the edge router and gateway is synchronised as a *consistently configured state* for that prefix. Whether a state 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 state space analysis was conducted in three steps starting with the simplest configuration of the protocol where no packet loss and no expiration of timers are allowed. The simplifications were then gradually lifted.

4.2 Step 1: Basic Configurations

The first step was to consider the simplest possible configurations of ERDP starting with a single prefix and assuming that there is no loss of packets on the wireless link and that prefixes do not expire. The full state space for this configuration has 46 nodes and 65 arcs and was generated in less than one second on a Linux PIII PC with 1Gb of memory. The SCC-graph has 36 nodes and 48 arcs. Having constructed the state space and computed the SCC-graph, the standard state space report can be generated. Inspection of the state space report showed that there was a single dead marking (a state without enabled transitions) represented by node 36. Hence the state represented by node 36 corresponds to a state where the protocol has terminated. Inspection of node 36 showed that it corresponded to a state where all the packet buffers were empty, but where the edge router and gateway are unsynchronised in the sense that the edge router is in a state where the gateway is assigned prefix P1 (the single prefix), but the gateway is not configured with that prefix. This is an error in the protocol. To locate the source of the problem, query functions in the state space tool were used to obtain a path (i.e., an error-trace or counter example) leading from the node representing the initial state to node 36. Figure 12 shows this error-trace visualised using a MSC. The integration of message sequence charts in the CPN computer tools makes it possible to automatically display a path in the state space as an MSC. The problem is that the edge router sends two unsolicited RA. The first one gets through and the gateway is configured with the prefix (Event A). However, when the second RS (Event B) without any prefixes is received by the edge router, 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 (Event C), and the gateway is no longer configured with a prefix.

To fix the error identified, the protocol was modified such 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 contains 34 nodes and 49 arcs, and there are no dead markings in the state space. The state space report specifies that there are 11 *home markings*. Hence, the protocol has the property that any of the corresponding 11 states can always be reached. Inspection of these 11 states showed that they all represented consistently configured states for the prefix P1. The states are contained in the single terminal SCC of the state space. When the SCC-graph has a single terminal SCC, it is always possible to reach one of the states in that terminal SCC and once having entered a state in the

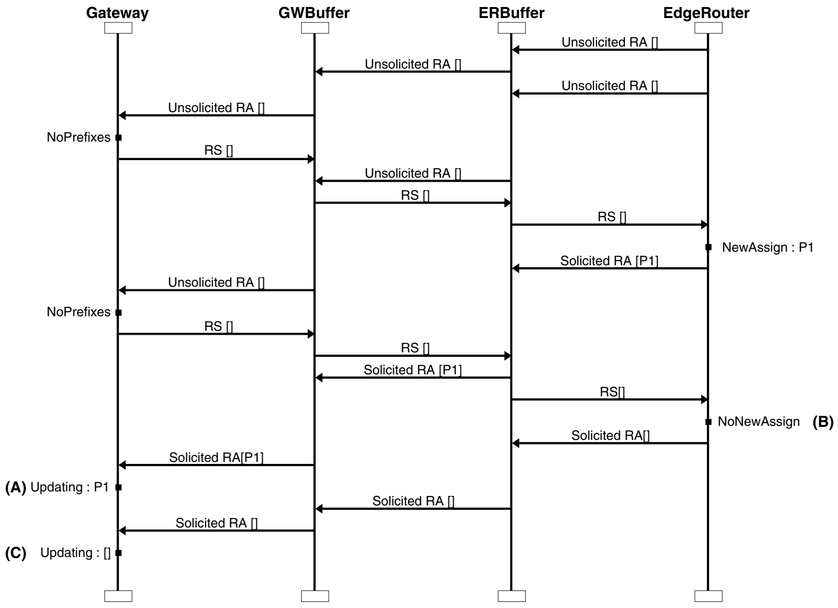


Fig. 12. MSC showing execution leading to undesired terminal state.

terminal SCC, the system can only enter states in the terminal SCC. This shows that when executing ERDP starting from the initial state it is always possible to reach a consistently configured state for the prefix, and when such a state has been reached, the protocol entities will remain in a consistently configured state. To verify that a consistently configured state will eventually be reached, it was checked that the single terminal SCC was the only non-trivial SCC. This shows that all cycles in the state space (which correspond to non-terminating executions of the protocol) are contained in the terminal SCC which (from above) contains 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. Since we ignore expiration of prefixes, the edge router will always refresh the prefix.

The number of prefixes was increased once the correctness of the protocol was established for a single prefix. When there is more than one prefix available it no longer holds that a state 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 may not eventually be reached. Instead it was verified that there was a single terminal SCC of which all states are states where all prefixes have been consistently configured. This shows that it is always possible to reach such a state, and when the protocol has consistently configured all prefixes, the protocol entities will

Table 2. State space statistics for basic configurations.

|P|	Nodes	Arcs	G-time	A-time
1	34	49	0.05	0.01
2	72	121	0.11	0.01
3	110	193	0.18	0.01
4	148	265	0.28	0.01
5	186	337	0.38	0.01
6	224	409	0.50	0.02
7	262	481	0.62	0.02
8	300	553	0.76	0.03
9	338	625	0.93	0.03
10	376	697	1.15	0.04

remain consistently configured. Secondly, it was checked that all states in each non-trivial SCC represented states where the protocol entities were consistently configured with a subset of the prefixes available in the edge router.

Table 2 lists the statistics for the state space generation for different number of prefixes. The |P| column specifies the number of prefixes. The Nodes and Arcs columns give the number of nodes and arcs in the state space, respectively. The G-time column gives the time in seconds used to generated the state space, and the A-time column gives the time in seconds used to conduct the verification of properties presented above. It can be observed that 38 states 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 the assignment of the first prefix. It can be seen that once the state space has been generated, the verification of properties could be done in less than one second.

4.3 Step 2: Adding Packet Loss

Next we considered state space analysis in presence of packet loss on the wireless link between the edge router and the gateway. First we considered the case where there is only a single prefix for distribution. The state space for this configuration has 40 nodes and 81 arcs. Inspection of the state space report showed that there was a single dead marking. This state represented an undesired terminal state where the prefix was assigned by the edge router according to its internal state, but the gateway was not configured with this prefix. Figure 13 shows an MSC corresponding to the path in the state space from the initial state to the undesired terminal state. The problem is that when the unsolicited RA containing the prefix is lost, the edge router will have assigned its last prefix and is no longer sending any unsolicited RAs. Furthermore, there are no timeouts in the protocol entities that can trigger the retransmission of the prefix to the gateway.

The problem identified above was fixed by ensuring that the edge router will resend an unsolicited RA to the gateway as long as it has prefixes assigned to the gateway according to its internal state. The state space of the revised CPN

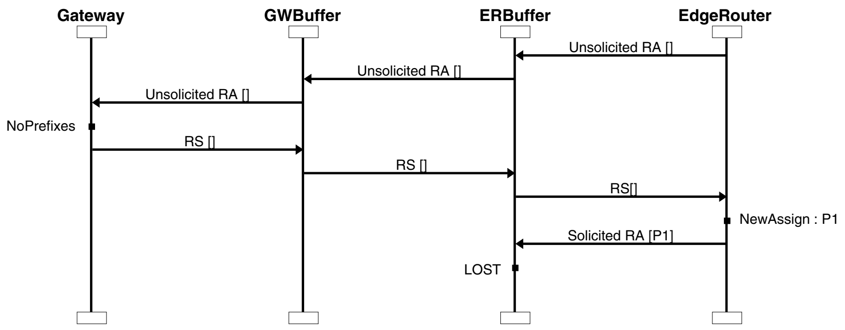


Fig. 13. MSC showing execution leading to undesired terminal state.

model has 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 states. 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 states in the undesired terminal SCC hence represent a livelock in the protocol, i.e., if one of the states 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 livelock was related to the control fields used in the router advertisements for refreshing prefixes and their interpretation in the gateway. This was identified by obtaining the MSC for a path leading from the initial state to one of the states 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 has 68 nodes and 160 arcs. The state space has a single terminal SCC containing 20 nodes which all represents states where the protocol entities are 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 were inspected using a query function to investigate the circumstances under which the protocol entities would not eventually be consistently configured. The query function 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 SCC 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 the absence of reaching a consistently configured state is due to packet loss and nothing else. Hence, if only finitely many packets are lost, a consistently configured state will *eventually* be reached. Table 3 gives statistics on the state space for verification of properties.

Table 3. State space statistics for packet loss configurations.

|P|	Nodes	Arcs	G-time	A-time
1	68	160	0.11	0.01
2	172	425	0.34	0.02
3	337	851	0.87	0.08
4	582	1489	1.48	0.16
5	926	2390	2.67	0.32
6	1388	3605	4.48	0.67
7	1987	5185	6.66	1.34
8	2742	7181	9.99	2.65
9	3672	9644	14.15	4.86
10	4796	12625	19.93	10.17

4.4 Step 3: Adding Expire of Prefixes

The final step in the analysis was to allow prefixes to expire. The analysis was conducted first in the configuration where the edge router has only a single prefix to distribute. The state space for this configuration has 173 nodes and 513 arcs. The state space has a single dead marking, and inspection of this dead marking showed that it corresponded to 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 is also a home marking, meaning that the protocol can always enter the expected terminal state.

When prefixes can expire the two protocol entities may never enter a consistently configured state. The reason is that a prefix may expire in the edge router (albeit unlikely) before the gateway has successfully been configured with the prefix. Hence, we are only able to prove that for any state where a prefix is still available in the edge router, it is possible to reach a state where the gateway and the edge router are consistently configured with this prefix. Table 4 lists the statistics for the state space generation and verification of properties in the case where expire of prefixes is also taken into account.

5 Conclusions

We have described how CPN modelling and state space analysis have been applied in the process of developing ERDP. Already the act of constructing the CPN model based on the ERDP specification provided valuable input to the ERDP specification, and the use of simulation added further insight into the operation of the protocol. State space analysis starting with the simplest possible configuration of the protocol identified additional errors in the protocol. The state space analysis succeeded in establishing the key properties of ERDP. The main drawback of verification based on state spaces is the state explosion

Table 4. State space statistics for prefix expire configurations.

|P|	Nodes	Arcs	G-time	A-time
1	173	531	0.34	0.02
2	714	2404	1.80	0.17
3	2147	7562	6.34	0.67
4	5390	19516	18.65	2.09
5	11907	43976	48.56	6.39
6	23905	89654	121.07	15.36
7	44550	169169	289.91	33.14
8	78211	300072	671.24	64.12
9	130732	505992	1560.73	123.81
10	209732	817903	3586.23	229.70

problem [28]. However, for the verification of ERDP presented in this paper the state explosion problem was not encountered and we succeeded in verifying the key properties of the ERDP protocol for the configurations that are envisioned to occur in practice. The verification presented in this paper considered the case with one gateway and one edge router. As part of future work we plan to consider verification in the presence of multiple gateways and edge routers. When considering multiple gateways and edge routers we are likely to encounter the state explosion problem. The symmetry method [19] and sweep-line method [2] are promising candidates for alleviating the state explosion problem in that case.

It can be argued whether or not the issues and errors discovered in process of modelling and conducting state space analysis would have been identified if additional conventional reviews of the ERDP specification had been conducted. Some of them probably would, but more subtle problems such as the synchronisation issues discovered during state space analysis would probably not have been discovered until a 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, and there are too many of these to do it in a systematic way. This demonstrates the value of being able to conduct state space exploration of the CPN model and in this way cover all execution sequences.

The construction of a CPN model can be seen as a very thorough and systematic way of reviewing a design specification of a protocol. Using an iterative process where both a conventional natural language specification and a CPN model is developed (as in this project) appears to be an effective way of integrating CPN modelling and analysis into the development of a protocol. In general, we believe that a combination of an executable formal model (such as a CPN model) and a natural language specification is a useful specification of a protocol. One reason that both are required is that the people who are going to implement the protocol may not be familiar with CP-nets. Secondly, there are important parts of the ERDP specification that are not reflected in the CPN model, such as the layout of packets. Similar observations were also made in [24] for the design of a security system.

In the project, we have demonstrated the use of message sequence charts (MSCs) for visualising the behaviour of the protocol based on simulations and based on paths in the state space. The automatic creation of MSCs based on executions of a CPN model is different from the conventional use of MSCs as a specification technique. The event-based graphical feedback in the form of MSCs has the advantage that it provides a compact view of the steps, i.e., how the current state of the protocol was reached. In contrast, the conventional state-based visualisation of the token distribution on the CPN model only shows the current state and it is distributed across several pages (modules). The use of MSCs in this project was of particular relevance since it presented the operation of the protocol in a form well-known to protocol developers.

We consider the application of CP-nets in the development of ERDP a success for three main reasons. Firstly and as in earlier case studies [16], we have demonstrated that the CPN modelling language and supporting computer tools are powerful enough to specify and analyse a real-world communication protocol. Secondly, the act of constructing the CPN model, executing, and discussing it lead to the identification of several non-trivial design errors and issues that under normal circumstances would not have been discovered until at best in the implementation phase. Finally, the effort of constructing the CPN model and conducting the state space analysis was approximately 100 man-hours. This is a relatively small investment compared to the many issues that were identified and resolved early as a consequence of constructing and analysing the CPN model.

Acknowledgements. The authors would like to acknowledge Nis A. Clausen, Rolf Christensen, and Shahrokk Sheik-El-Ameh from Ericsson Telebit A/S for their significant contributions in the project. The work on the design and specification of ERDP has been carried out on the Building Blocks for Network Centric Warfare (B2NCW), EUCLID/Eurofinder programme, Project RTP6.22, Contract No. 02/EF 6.22/012, with support from the UK, Swedish and Danish MoDs and the participating companies. The authors are grateful for this support and would like to acknowledge the contributions of all the B2NCW team members.

References

1. S. Christensen. *Message Sequence Charts. User's Manual*, January 1997. Available via `www.daimi.au.dk/designCPN`.
2. S. Christensen, L.M. Kristensen, and T. Mailund. A Sweep-Line Method for State Space Exploration. In *Proc. of 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of *Lecture Notes in Computer Science*, pages 450–464. Springer-Verlag, 2001.
3. E. M. Clarke and J. M. Wing. Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, 28(4):626–643, 1996.
4. A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPV6) Specification, December 1998. RFC 2463. Work in progress.
5. CPN Tools. `www.daimi.au.dk/CPNtools`.

6. The CPN Group at University of Aarhus. `www.daimi.au.dk/CPnets`.
7. S. Deering and R. Hinden. Internet Protocol, Version 6 (IPV6) Specification, December 1998. RFC 2460. Work in progress.
8. J. Desel. Basic Linear Algebraic Techniques for Place/Transition Nets. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 257–308. Springer-Verlag, 1998.
9. J. Desel. Validation of Process Models by Construction of Process Nets. In *Business Process Managements - models, techniques and empirical studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 110–128. Springer-Verlag, 2000.
10. J. Desel and W. Reisig. Place/Transition Petri Nets. In *Lecture on Petri nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 122–173. Springer-Verlag, 1998.
11. Design/CPN. `www.daimi.au.dk/designCPN`.
12. Ericsson Telebit A/S. `www.tbit.dk`.
13. J. Esparza. Model Checking using Net Unfoldings. *Science of Computer Programming*, 23:151–195, 1994.
14. Internet Engineering Task Force. Mobile ad-hoc networks. `www.ietf.org/html.charters/manet-charter.html`.
15. C. Huitema. *IPv6: The New Internet Protocol.* Prentice-Hall, 1998.
16. Examples of Industrial Use of CP-nets. `www.daimi.au.dk/CPnets/intro/example_indu.html`.
17. The Internet Engineering Task Force. `www.ietf.org`.
18. K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use. Vol. 1-3.* Springer-Verlag, 1992-1997.
19. K. Jensen. Condensed State Spaces for Symmetrical Coloured Petri Nets. *Formal Methods in System Design*, 9(1/2):7–40, 1996.
20. L.M. Kristensen. Ad-hoc Networking and IPv6: Modelling and Validation. `www.pervasive.dk/projects/IPv6/IPv6_summary`.
21. L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.
22. T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6), December 1998. RFC 2461. Work in progress.
23. C.E. Perkins. *Ad Hoc Networking.* Addison-Wesley, 2001.
24. J.L. Rasmussen and M. Singh. Designing a Security System by Means of Coloured Petri Nets. In *Proc.of 17th International Conference on Application and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 400–419. Springer-Verlag, 1996.
25. W. Reisig. *Petri Nets*, volume 4 of *EACTS Monographs in Theoretical Computer Science.* Springer-Verlag, 1985.
26. M. Silva, E. Teruel, and J. M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 309–373. Springer-Verlag, 1998.
27. J.D. Ullman. *Elements of ML Programming.* Prentice-Hall, 1998.
28. A. Valmari. The State Explosion Problem. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer-Verlag, 1998.