Chapter 8

Model-based Prototyping of an Interoperability Protocol for Mobile Ad-hoc Networks

The paper *Model-based Prototyping of an Interoperability Protocol for Mobile Ad-hoc Networks* presented in this chapter has been published as a conference paper [T4].

[T4] L.M. Kristensen, M. Westergaard, and P.C. Nørgaard. Model-based Prototyping of an Interoperability Protocol for Mobile Ad-hoc Networks. In *Proc. of IFM'05*, volume 3771 of *LNCS*, pages 266–286. Springer-Verlag, 2005.

The version presented here is identical to the conference paper except for minor typographical changes.

Model-based Prototyping of an Interoperability Protocol for Mobile Ad-hoc Networks

L. M. Kristensen*† M. Westergaard* P. C. Nørgaard‡

*Department of Computer Science, University of Aarhus,
IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark,
Email: {kris,mw}@daimi.au.dk

‡Ericsson Danmark A/S, Telebit,
Skanderborgvej 222, DK-8260 Viby J, Denmark,
Email: Peder.Chr.Norgaard@ericsson.com

Abstract

We present an industrial project conducted at Ericsson Danmark A/S, Telebit where formal methods in the form of Coloured Petri Nets (CP-nets or CPNs) have been used for the specification of an interoperability protocol for routing packets between fixed core networks and mobile ad-hoc networks. The interoperability protocol ensures that a packet flow between a host in a core network and a mobile node in an ad-hoc network is always relayed via one of the closest gateways connecting the core network and the mobile ad-hoc network. This paper shows how integrated use of CP-nets and application-specific visualisation have been applied to build a model-based prototype of the interoperability protocol. The prototype consists of two parts: a CPN model that formally specifies the protocol mechanisms and a graphical user interface for experimenting with the protocol. The project demonstrates that the use of formal modelling combined with the use of application-specific visualisation can be an effective approach to rapidly construct an executable prototype of a communication protocol.

Keywords: Model-driven prototyping; animation; Coloured Petri Nets; mobile ad-hoc network.

8.1 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 protocols a challenging task. Protocols operating in networks with mobile nodes and wireless communication present an additional set of challenges in protocol engineering since the orchestration of realistic scenarios with many mobile nodes is impractical, and the physical characteristics of wireless communication makes reproduction of errors and scenarios almost impossible.

†Supported by the Danish Natural Science Research Council.

We present a case study from a joint research project [101] between the Coloured Petri Nets Group [34] at University of Aarhus and Ericsson Danmark A/S, Telebit [47]. The research project applies formal methods in the form of Coloured Petri Nets (CP-nets or CPNs) [91, 102] and the supporting CPN Tools [33] in the development of Internet Protocol Version 6 (IPv6) [82] based protocols for ad-hoc networking [137]. 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.

CP-nets is a graphical discrete-event modelling language applicable for concurrent and distributed systems. CP-nets are based on Petri nets [143] and the programming language Standard ML (SML) [159]. Petri nets provide the foundation of the graphical notation and the basic primitives for modelling concurrency, communication, and synchronisation. The SML programming language provides the primitives for the definition of data types, modelling data manipulation, and for creating compact and parameterisable models. CPN models are executable and describe the states of a system and the events (transitions) between the states. CP-nets includes a module concept that makes it possible to organise large models into a hierarchically related set of modules. The CPN modelling language is supported by CPN Tools and have previously been applied in a number of projects for modelling and validation of protocols (see, e.g., [61, 64, 103, 132]).

The use of formal modelling languages such as CP-nets for specification and validation of protocols is attractive for several reasons. One advantage of formal models is that they are based on the construction of executable models that make it possible to observe and experiment with the behaviour of the protocol prior to implementation using, e.g., simulation. This typically leads to more complete specifications since the model will not be fully operational until all parts of the protocol have been at least abstractly specified. A model also makes it possible to explore larger scenarios than is practically possible with a physical setup. Another advantage of formal modelling is the support for abstraction, making it possible to specify protocols while ignoring many implementation details.

From a practical protocol engineering viewpoint, the use of formal modelling also have some shortcomings. Even if the modelling language supports abstraction and a module concept there is in most cases an overwhelming amount of detail in the constructed model. This is a disadvantage, in particular when presenting and discussing the design with colleagues unfamiliar with the applied modelling language. This means that a formal specification in many cases is accompanied by informal drawings being developed in parallel. The level of detail can also be a disadvantage when exploring the protocol design via, e.g., simulation. Furthermore, even if a model is executable, it still lacks the application- and domain-specific appeal of a conventional prototype.

The contribution of this paper is to present a model-based prototyping approach where formal modelling is integrated with the use of an animation GUI for visualising system behaviour to address the shortcomings of formal modelling discussed above. The approach has been applied to an interoperability protocol for routing packets between nodes in a mobile ad-hoc network and hosts in a fixed core network. Formal modelling is used for the specification of the protocol mechanisms and an application- and domain-specific GUI [C2] is added on top of the CPN model. The result is a *model-based prototype* in which

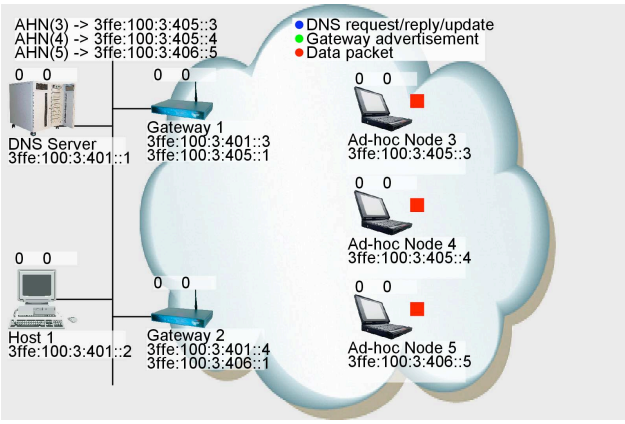

Figure 8.1: The hybrid network architecture.

the animation GUI makes it possible to observe the behaviour of the system and provide stimuli to the protocol. The use of an underlying formal model is fully transparent when experimenting with the prototype. The animation GUI has been used in the project both internally during protocol design and externally when presenting the designed protocol to management and protocol engineers not familiar with CPN modelling.

The rest of the paper is organised as follows. Section 8.2 gives a brief introduction to the network architecture and the interoperability protocol, and Sect. 8.3 presents the model-based prototyping approach. Section 8.4 presents selected parts of the CPN model specifying the interoperability protocol. Section 8.5 presents the graphical animation user interface and package applied in the project. Finally, Sect. 8.6 sums up the conclusions and presents related work.

8.2 The Interoperability Protocol

Figure 8.1 shows the hybrid network architecture captured by the model-based prototype. The network architecture consists of two parts: an IPv6 core network (left) and a mobile ad-hoc network (right). The core network consists of a Domain Name System (DNS) Server and Host 1. The mobile ad-hoc network contains three mobile nodes (Ad-hoc Node 3-5). The core network and the mobile ad-hoc network are connected by Gateway 1 and Gateway 2. A routing protocol for conventional IP networks (such as OSPF [110]) is deployed in the core network and a routing protocol for ad-hoc networks (such as OLSR [29]) is used in the mobile ad-hoc network. The purpose of the interoperability protocol is to ensure that packets are routed between hosts in the core network and nodes in the mobile ad-hoc network via the closest gateway.

The gateways periodically announce their presence to nodes in the mobile ad-hoc network by sending *gateway advertisements* containing an IPv6 *address prefix*. The address prefixes announced by the gateways are assumed to be unique, and the advertisement can be distributed to the ad-hoc nodes using, e.g., flooding. The interoperability protocol does not rely on a specific dissemination mechanism for the gateway advertisements. The interoperability protocol generalises to an arbitrary number of gateways and mobile nodes. Figure 8.1 shows the concrete setup represented in the model-based prototype.

IPv6 addresses [69] 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*. 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 network architecture depicted in Figure 8.1, Gateway 1 is announcing the 64-bit address prefix 3ffe:100:3:405::/64 and Gateway 2 is announcing the prefix 3ffe:100:4:406::/64. 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 Gateway 1, whereas Ad-Hoc Node 5 has configured an IP address based on the prefix announced by Gateway 2. For an example, Ad-hoc Node 3 has configured the address 3ffe:100:3:405::3.

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. Gateway 1 has configured the address 3ffe:100:3:405::1 and Gateway 2 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. Host 1 in the core network has configured the address 3ffe:100:3:401::2 and the DNS server has configured the address 3ffe:100:3:401::1. The ad-hoc nodes may receive advertisements from both gateways and configure an IPv6 address based on each of the prefixes. The reachability of the address prefixes announced by the gateways in the ad-hoc network are announced in the core network via the routing protocol executed in the core network.

The basic idea in the interoperability protocol is that the mobile nodes register the IPv6 address in the DNS database which corresponds to the preferred (closest) gateway. Updates to the DNS database relies on the Dynamic Domain Name System Protocol [168]. The entries in the DNS database related to the mobile nodes are shown to the upper left in Figure 8.1. For an example, it can be seen that the entry for Ad-hoc Node 3 (AHN(3)) is mapped to the address 3ffe:100:3:405::3. When a mobile ad-hoc node discovers that another gateway is closer, it will send an update to the DNS server causing its DNS entry to be changed to the IPv6 address based on the prefix announced by the new gateway. It is assumed that the routing protocol executed in the mobile ad-hoc network will provide the information required for a mobile node to determine its distances to the currently reachable gateways. This means that when Host 1 wants to communicate, with e.g., Ad-hoc Node 3 and makes a DNS request to resolve the IP address of Ad-hoc Node 3, the DNS server will return the IP address corresponding to the prefix announced by the gateway closest to Ad-hoc Node 3.

8.3 Model-based Prototyping Methodology

Figure 8.2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Figure 8.2) has been developed by modelling the natural language protocol specification [130] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Figure 8.1 and the protocol mechanisms of the interoperability protocol, e.g., the periodic transmission of advertisements, the dynamic updates

of the DNS database, and traffic flows between hosts in the core network and nodes in the ad-hoc network. The resulting model can already be viewed as an early prototype since it is possible to execute and experiment with the protocol at the level of the CPN model. Since CP-nets is a graphical modelling language, it is possible to observe the execution of the model directly on the CPN model.

The CPN model provides a very detailed view on the execution of the system and it can be an advantage to provide a high-level way of interacting and experimenting with the prototype. Furthermore, when presenting the protocol design to people not familiar with CP-nets, it can be an advantage to be able to demonstrate the prototype without directly relying on the CPN model but more application and domain specific means. To support this, an *animation GUI* (top left of Figure 8.2) has been added on top of the CPN model. This graphics visualises the execution of the prototype using the graphical representation of the network architecture previously shown in Figure 8.1. The graphics is updated by the underlying CPN model according to the execution of the protocol.

In addition to observe feedback on the execution of the system in the animation GUI, it is also possible to provide input to the system directly via the animation GUI. In the prototype, it is possible for the demonstrator (e.g., a protocol engineer) to move the nodes in the ad-hoc network and to define traffic flows from the host in the core network to the nodes in the mobile ad-hoc network. The animation GUI has been implemented using a general visualisation package and framework [C2] developed in the course of the project (see Sect. 8.4).

Altogether the approach makes it possible to explore and demonstrate the prototype of the interoperability protocol based on the CPN model that formally captures the design, but doing it in such a way that the use an underlying formal model is transparent for the observer and the demonstrator.

8.4 The CPN Model

This section presents the CPN model specifying the interoperability protocol. The complete CPN model is hierarchically structured into 18 modules. As the CPN model is too large to be presented in full in this paper, we present only selected parts of the CPN model. The aim is to show how the key aspects of the interoperability protocol have been modelled. The key concepts of CP-nets

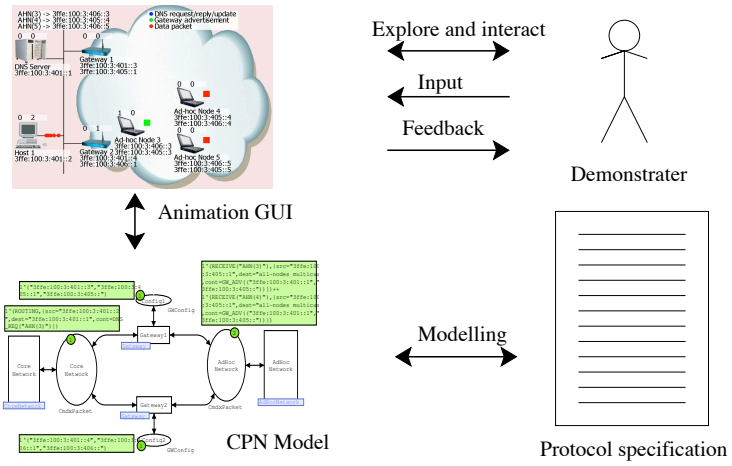

Figure 8.2: Model-based prototyping approach.

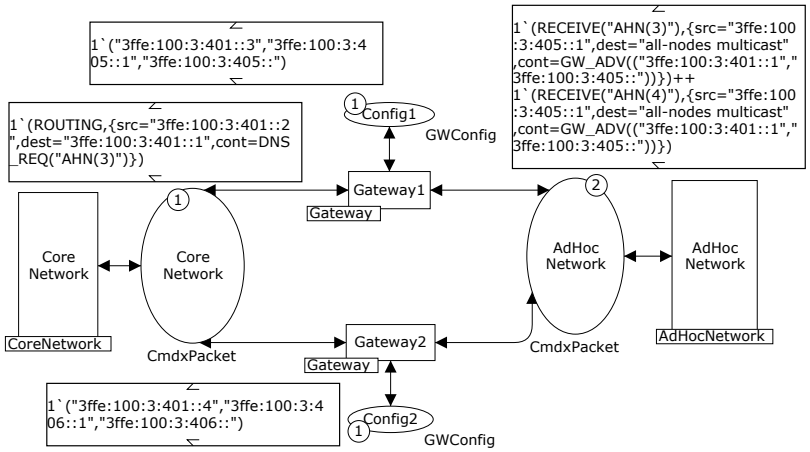

Figure 8.3: System module – top-level module of the CPN model.

will be briefly introduced as we proceed with the presentation. The reader is referred to [102] for a comprehensive introduction to CP-nets.

8.4.1 Model Overview

The module concept of CP-nets is based on the notion of *substitution transitions* which have associated *submodules* describing the compound behaviour represented by the substitution transition. A submodule of a substitution transition may again contain substitution transitions with associated submodules. Figure 8.3 shows the top level module of the CPN model which is composed of three main parts represented by the rectangular substitution transitions CoreNetwork (left), Gateway1 and Gateway2 (middle), and AdHocNetwork (right). The substitution transition CoreNetwork and its submodules model the core network, the substitution transition AdHocNetwork and its submodules model the mobile ad-hoc network, and the submodules of the two Gateway substitution transitions model the operation of the gateways connecting the core network and the mobile ad-hoc network. The text in the small rectangular box attached to each substitution transition gives the name of the associated submodule.

The state of a CPN model is represented through *places* (drawn as ellipses). There are four places in Fig. 8.3. The places CoreNetwork and AdHocNetwork are used for model modelling the packets in transit on the core network and ad-hoc network, respectively. The state of a CPN model is a distribution of tokens on the places of the CPN model. Figure 8.3 depicts a state where there is one token on place CoreNetwork and two tokens on place AdHocNetwork. The number of tokens on a place is written in the small circle attached to the place. The *data values* (colours) of the tokens are given in the filled box positioned next to the small circle. As an example, place CoreNetwork contains one token with the colours:

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

representing a DNS request in transit from Host 1 to the DNS server. Place AdHocNetwork contains two tokens representing gateway advertisements in transit to nodes in the ad-hoc network. The two Config places each contains a token representing the configuration of the corresponding gateway. It consists

of the IP address of the interface connected to the core network, the IP address of the interface connected to the ad-hoc network, and the prefix announced.

The data values (colours) of tokens that can reside on a place are determined by the *colour set* of the place which by convention is written below the place. Colour sets are similar to types known from conventional programming languages. Figure 8.4 lists the definitions of the colour sets (types) 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 the interoperability protocol are modelled by PacketCont:

DNS_REQ modelling a DNS request packet. It contains the symbolic IP address to be resolved.

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 transmitted between hosts and the mobile nodes.

The colour set `Packet` models the packets as a record containing the source, destination, and the content. The actual payload (content) and layout of packets are indifferent 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.

8.4.2 Modelling the Core Network

Figure 8.5 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 Figure 8.3. The *port place* CoreNetwork is assigned to the CoreNetwork *socket place* in the System module (see Figure 8.3). Port places are indicated by the In, Out, or I/O tags associated with them. The assignment of a port place to a socket place means that the two places are linked together and will always have identical tokens. By adding and removing tokens from port places, it is possible for a submodule to exchange tokens with its environment. 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.

Hosts.

Figure 8.6 depicts the Host module modelling the host on the core network. The port place CoreNetwork (bottom) is assigned to the CoreNetwork socket place in the CoreNetwork module (see Figure 8.5). The module models the transmission of packets from the host to one of the mobile ad-hoc nodes. The substitution

```
(* --- Addressing --- *)
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;

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

(* --- Gateways configuration --- *)
colset GWConfig = product IPAdr * IPAdr * Prefix;
```

Figure 8.4: Colour set definitions used in the System module.

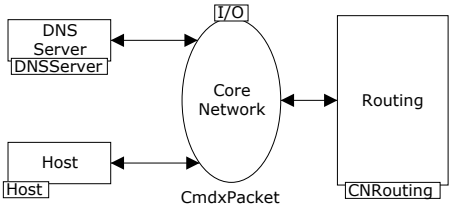

Figure 8.5: Core Network module – modelling the core network.

transition Flows (top) is used for interfacing with the animation GUI. We will return to this issue in Sect. 8.5.

The remaining places and transitions are used for modelling the behaviour of the host. The rectangles in Fig. 8.6 are ordinary transitions (i.e., not substitution transitions) which means that they can become *enabled* and *occur*. The dynamics of a CPN model consists of occurrences of enabled transitions that change the distribution of tokens on the places. An occurrence of a transition removes tokens from places connected to incoming arcs of the transition and adds tokens to places connected to outgoing arcs of the transition. The colours of the tokens removed from input places and added to output places are determined by *evaluating* the *arc expressions* on the arcs surrounding the transition. The arc expressions are written in the SML programming language. Data val-

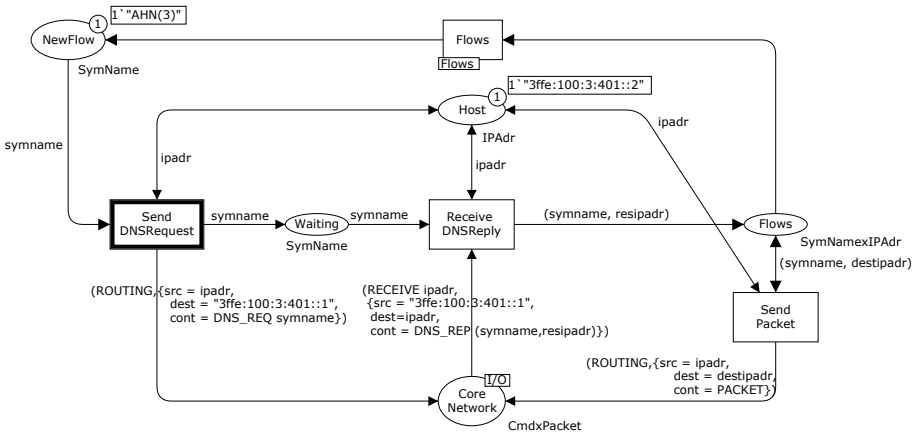

Figure 8.6: Host module – modelling the host.

ues must be bound to the *variables* appearing in the surrounding arc expressions before the arc expressions can be evaluated. This is done by creating a *binding element* which is a pair (t, b) consisting of a transition t and a *binding* b assigning data values to the variables of the transition. A binding element (t, b) is enabled iff the multi-set of tokens obtained by evaluating each input arc expression is a subset of the tokens present on the corresponding input place.

When the user defines a flow in the animation GUI, a token will appear in place NewFlow with a colour corresponding to the symbolic name of the mobile ad-hoc node which is the destination of the packet flow. An example is given in Fig. 8.6, where the NewFlow place contains a token corresponding to the user having defined a flow to Ad-hoc Node 3. This enables the SendDNSRequest transition in a binding where the value `"AHN(3)"` is bound to the variable `symname` of type SymName and the variable `ipadr` is bound to the value of the token on place Host specifying the IP address of the host.

When the SendDNSRequest transition occurs in the above binding, it will remove tokens from places `NewFlow` and `Host`, and add tokens to the output places Host, Waiting, and CoreNetwork. Tokens are added to the Host place since SendDNSRequest and Host are connected by a double arcs which is a short-hand for an arc in each direction having identical arc expressions. The colour of the tokens added are determined by evaluating the expressions on the output arcs. The resulting state is shown in Fig. 8.7. A token representing the IP address of the host is put back on place Host, a token representing the symbolic name to be resolved is put on place Waiting, and a token representing a DNS request has been put on place CoreNetwork.

The reception of the DNS reply from the DNS server is modelled by the transition ReceiveDNSReply which causes the token on place Waiting to be removed and a token to be added on place Flows. This corresponds to the host entering a state in which packets can be transmitted to the mobile ad-hoc node. The sending of packets is modelled by the transition SendPacket. The user may then decide (via the animation GUI) to terminate the packet flow which will cause the token on place Flows to be removed, and transmission of packets will cease. A host can have concurrent flows to different mobile ad-hoc nodes.

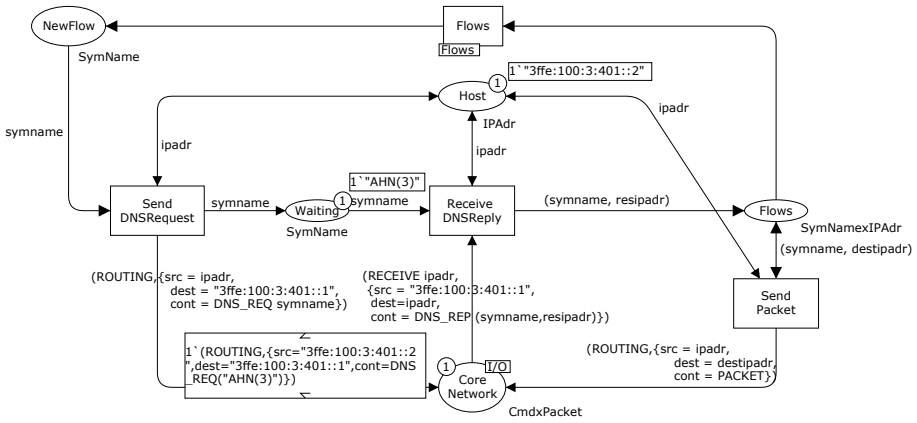

Figure 8.7: Host module – after occurrence of SendDNSRequest transition.

Domain Name Server and Database.

Figure 8.8 shows the DNSServer module modelling the DNS Server. The place DNSAdr contains a token corresponding to the IP address of the DNS Server. Place DNSDatabase models the DNS database entries on the DNS Server. There is a token on place DNSDatabase for each entry in the DNS database. The entries in the DNS database are modelled as tuples where the first component is the symbolic address (name) and the second component is the IP address bound to the symbolic name in the first component.

There are two possible events in the DNS server modelled by the transitions DNSRequest and DNSUpdate. The transition DNSRequest models the reception of DNS requests (from hosts) and the sending of the DNS reply containing the resolved IP address. The transition DNSUpdate models the reception of DNS updates from the mobile ad-hoc nodes. Both transitions access the DNSDatabase for lookup (transition DNSRequest) and modification (transition DNSUpdate).

Core Network Routing.

The CPN model does not specify a specific routing protocol but only the requirements to the core network routing protocol. This means that any routing protocol that meets these requirements can be used to implement the interop-

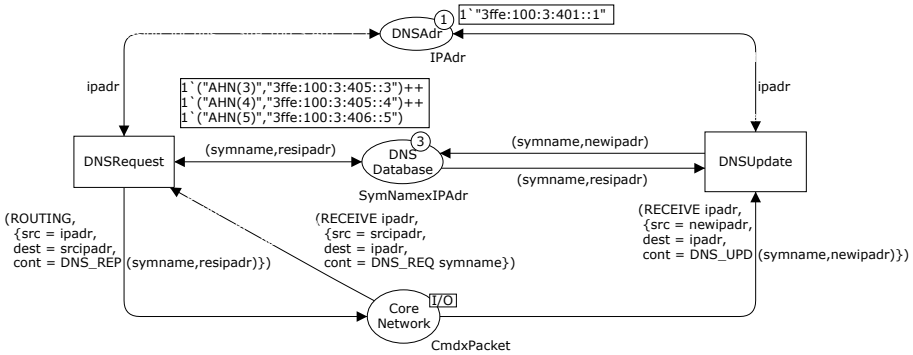

Figure 8.8: DNSServer module – modelling the DNS Server.

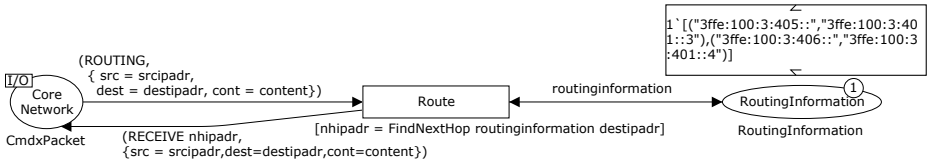

Figure 8.9: CNRouting module – Routing in the core network.

erability protocol. The routing mechanism in the core network is abstractly modelled by the CNRouting module shown in Figure 8.9. The place RoutingInformation models the routing information computed by the specific routing protocol in operation. This place contains a token that makes it possible given a prefix, to find the IP address of the corresponding gateway on the core network. This specifies the requirement that the gateways are required to participate in the routing protocol of the core network and announce a route to the prefix that they are advertising in the mobile ad-hoc network. This enables packets for nodes in the mobile ad-hoc network to be routed via the gateway advertising the prefix that matches the destination IP address of the packet. 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 function FindNextHop in the *guard expression* of the transition computes the IP address of the next hop gateway using the routing information and destination IP address of the packet.

8.4.3 Modelling the Gateways

The role of the gateway is to relay packets between the core network and the mobile ad-hoc network, and to periodically send advertisements to the mobile ad-hoc network. Figure 8.10 shows the Gateway module modelling the operation of the gateways. This module is the submodule of the two substitution transitions Gateway1 and Gateway2 on the System module. This means that there will be two *instances* of the Gateway module - one for each of the substitution transitions. Figure 8.10 shows the instance corresponding to Gateway1. The port place CoreNetwork is assigned to the socket place CoreNetwork and the port place AdHocNetwork is assigned to the socket place AdHocNetwork on the System module. The place Config contains a token giving the configuration of the gateway.

The relay of packets from the core network to the mobile ad-hoc network is modelled by the transition AHN_CoreTransmit and the relay of packets from the mobile ad-hoc network to the core network is modelled by the transition Core_AHNTransmit. Packets to be transmitted from the core network to the ad-hoc network are represented by tokens in the place CoreNetwork. When the transition Core_AHNTransmit occurs corresponding to the relay of a packet from the core network to the ad-hoc network, this token will be removed from the CoreNetwork place and a new token representing the packet added to the place AdHocNetwork. The relay of packets from the AdHocNetwork to the CoreNetwork is modelled in a similar manner by the transition AHN_CoreTransmit. The periodic transmission of advertisements on the mobile ad-hoc network is modelled by the substitution transition GatewayAdvertisement. The presentation of the submodule associated with this substitution transition has been omitted.

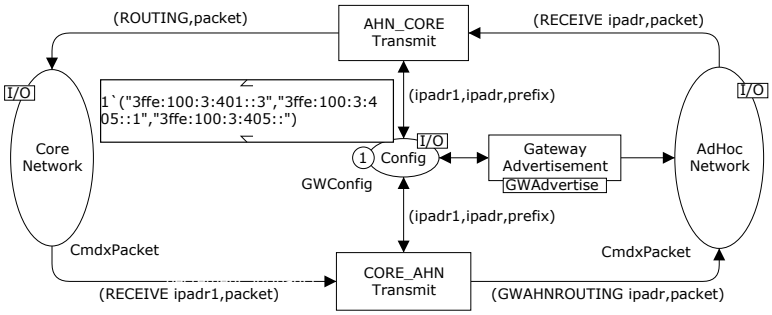

Figure 8.10: Gateway module – modelling the operation of the gateways.

8.4.4 Modelling the Mobile Ad-hoc Network

Figure 8.11 depicts the AdHocNetwork module which is the top level module of the part of the CPN model 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 the 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.

Figure 8.12 lists the definition of the colour sets used in the AdHocNetwork module. The topology of the mobile ad-hoc network is abstractly represented by only representing the distance from each of the ad-hoc nodes to the two gateways. The reason is that it is only the relative distance to the two gateways which are of relevance to the operability protocol – not the complete topology. The colour set DistanceInformation is used to keep track of the reachability between the nodes in the ad-hoc network and the gateways. The distance information is a list with an entry for each pair of ad-hoc node and gateway. Each entry is again list consisting of a four-tuple (colour set DistanceEntry). Each entry consists of the symbolic name of the mobile ad-hoc node, its IP address (if configured), the IP address of the gateway (if configured), and the distance to the gateway. The gateway may also be unreachable in which case the distance is set to NOTREACH.

The colour set AHNConfig is used to model the configuration information

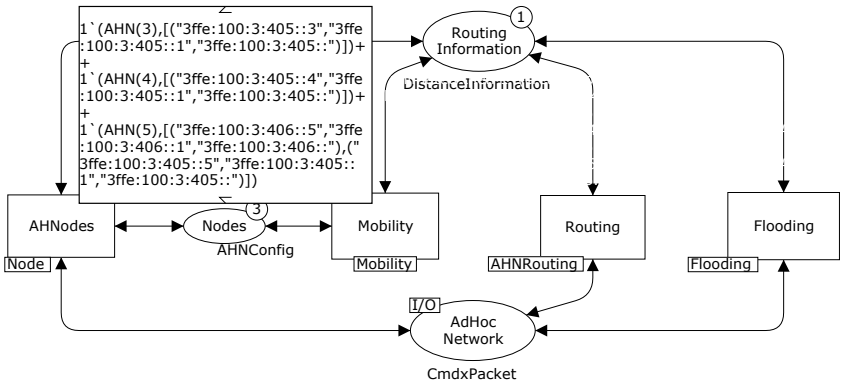

Figure 8.11: AdHocNetwork module – modelling the ad-hoc network.

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

(* --- distance information --- *)
colset Distance = union REACH : Dist + NOTREACH;
colset DistanceEntry = product AHNode * IPAdr * IPAdr * Distance;
colset DistanceInformation = list DistanceEntry;

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

Figure 8.12: Colour definitions used in the AdHocNetwork module.

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 of an IP address 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 node will ensure that the DNS database always contains the IP address corresponding to the *preferred gateway*.

There are four substitution transitions in the AdHocNetwork module corresponding to the components of the ad-hoc network represented:

AHNodes represents the behaviour of the nodes in the mobile ad-hoc network. This will be presented in more detail below.

Mobility represents the mobility of nodes in the ad-hoc network, i.e., that the nodes may move closer or further away from the gateways. We will return to the modelling of mobility in Sect. 8.5.

Routing represents the routing protocol executed in the ad-hoc network. The purpose of the routing protocol in the context of the interoperability 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.

Flooding models the dissemination of advertisements from the gateways. A detailed presentation of this part of the model has been omitted.

Figure 8.13 depicts the Node module specifying the operation of the ad-hoc nodes. The module has three substitution transitions. PacketReceive represents the reception of packets from hosts in the core network. The submodule PacketReceive of this substitution transition is shown in Figure 8.14. The transition PacketReceive models the reception of a packet and consumes the token on place AdHocNetwork corresponding to the packet being received. AdvReceive represents the reception of advertisements from the gateways. A node changes its preferred IP address if the received advertisement is from a gateway which is closer than the gateway corresponding to the currently preferred gateway (if any). If the node configures a new preferred IP address based on the received advertisement, then it will send an update to the DNS server containing the new preferred IP address. DeleteGW represents the case where the

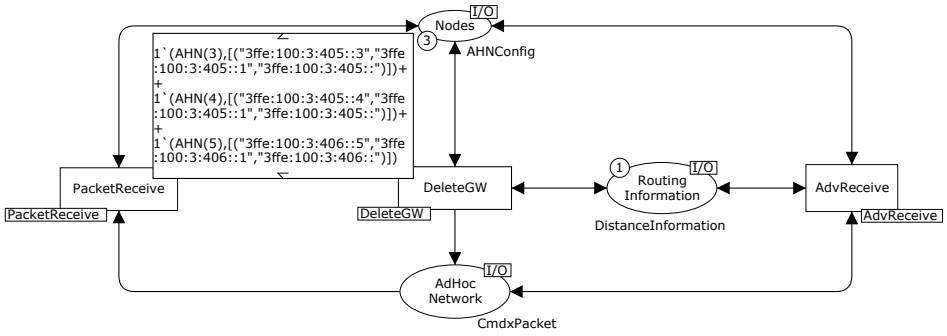

Figure 8.13: Node module – modelling an ad-hoc node.

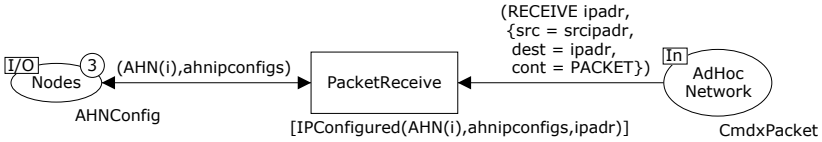

Figure 8.14: PacketReceive module – modelling reception of payload packets.

gateway corresponding to a configured IP address becomes unreachable. The assumption is that this will be detected via the routing protocol executed in the ad-hoc network or if advertisement has not been received for a specified amount of time. The submodules of the AdvReceive and DeleteGW are similar in complexity as the submodule of the PacketReceive substitution transition in Fig. 8.14 and has been omitted.

8.5 The Animation Graphical User Interface

The animation GUI has been implemented based on a general animation package [C2] developed in the course of the project. The animation package provides a general framework for adding various diagram types on top of executable models. The animation package is not designed specifically for CPN models, but is applicable also to other modelling formalisms.

The architecture of the model-based prototype developed in the project is depicted in Fig. 8.15 and consists of three main parts: The CPN Tools GUI (left), the CPN simulator (middle), and the animation GUI (right). The CPN Tools GUI and the CPN simulator constitute the CPN computer tools used in the project. CPN models are constructed using the CPN Tools GUI and the CPN simulator implements the formal semantics of CP-nets for execution of CPN models. The simulator communicates via the XML-RPC [170] infrastructure with the animation GUI to display the execution of the CPN model using the domain-specific graphics and for receiving stimuli/input from the demonstrator. The specific visualisation means are determined by the set of animation plug-ins used in the animation GUI. One animation plug-in was used to obtain *interaction graphics* in the form shown in Fig. 8.16. A second animation plug-in was used to obtain feedback in the form of message sequence charts (MSCs).

Figure 8.16 shows a representative snapshot of the application-specific 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.

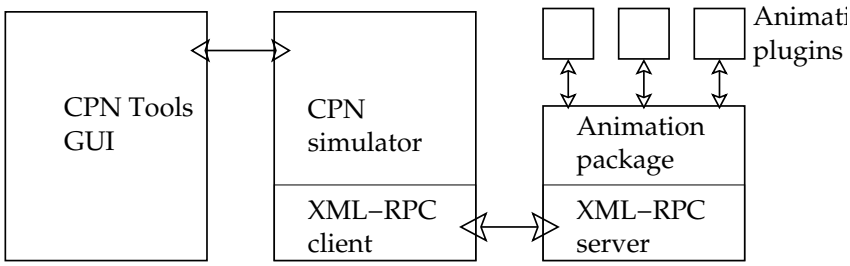

Figure 8.15: Architecture of the model-based prototype.

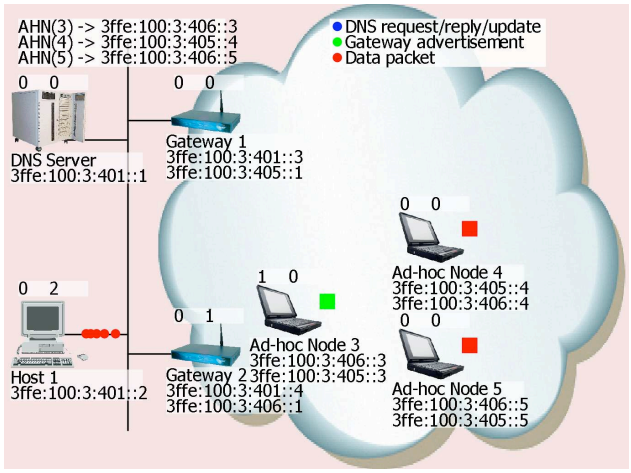

Figure 8.16: Snapshot of the interaction graphics.

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. Transmission of payload packets are visualised using read dots, and DNS packets are visualised using blue dots. Figure 8.16 shows an example where Host 1 is transmitting a payload packet to Ad-hoc Node 3.

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 flow from the host in the core network to one of the nodes in the mobile ad-hoc network by clicking on the read 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.

Figure 8.17 shows an example of a MSC 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.

Graphical feedback from the execution of the CPN model is achieved by

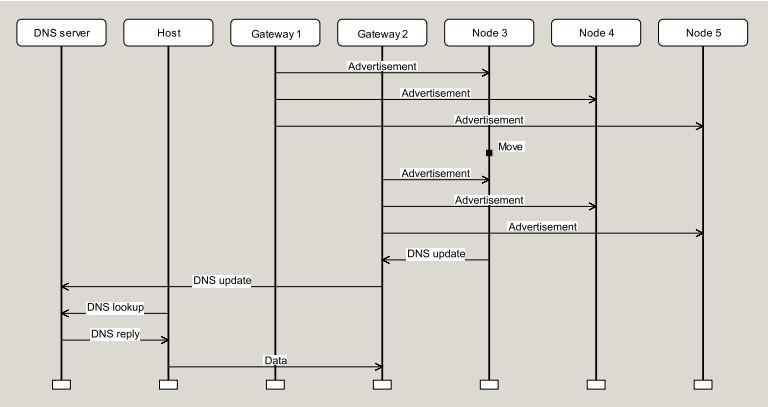

Figure 8.17: Message sequence chart generated by the animation GUI.

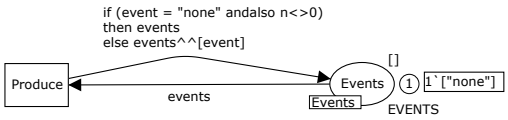

Figure 8.18: Poll module – Polling the animation GUI for events.

attaching *code segments* to the transitions in the CPN model. These code segments are executed whenever the corresponding transition occurs in the simulation/execution of the CPN model. As an example, the transition Route (see Figure 8.9) has an attached code segment which invokes the primitives required 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. The Poll module shown in Figure 8.18 polls the animation GUI for events at regular intervals during the execution of the CPN model and puts events into a list on the place Events, thereby implementing an event queue between the animation GUI and the CPN model. Parts of the CPN model that is to react on events from the animation GUI are linked to the Event place and are able to consume events from the event queue. The transition Produce polls the animation GUI for events.

8.6 Conclusions

We have presented our model-based prototype approach and demonstrated its use on an interoperability protocol. In addition to providing a detailed specification of the interoperability protocol via the constructed CPN model, the work has also highlighted the following characteristics and aspects of a model-based (virtual) prototyping approach:

Representation. 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 package in, e.g., JAVA, totally detached from the CPN model. We would then have obtained a model closer to the actual implementation. The disadvantage of this approach would have been a double representation of the dynamics of the interoperability protocol.

Transparency. The use of a domain specific graphical user interface (the animation GUI) has the advantage that the design can be experimented with and explored without having knowledge of the CPN modelling language. This has been shown in practise at a demonstration to management with no CPN knowledge.

Controllability. A model-based prototype is easier to 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.

Abstraction. Implementation details can be abstracted away and only the key part of the design have to be specified in detail. As an example, in the CPN model of the interoperability protocol we have abstracted away the routing mechanisms in the core and ad-hoc networks, and the mechanism used for distribution of advertisements. Instead, we have modelled the service provided by these components. The possibility of making abstraction means that it is possible to obtain an executable prototype without implementing all components.

Feasibility. 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. This also makes it possible to investigate larger scenarios, e.g., scenarios that may not be feasible to investigate with the available physical equipment.

Related Work

Integrated use of visualisation and formal modelling has also been considered for CP-nets in earlier work in the area of embedded systems [142], telecommunication protocols [14], pervasive electronic patient records [10], and software for mobile phones [112]. The case studies in [10, 14, 112, 142] all applied the MIMIC/CPN [141] package, an internal part of the DESIGN/CPN [37] tool. The approach presented in this paper relies on an external application handling the visualisation, which we find is a more flexible approach as it allows us to use existing software libraries supporting different diagram types. In MIMIC/CPN, input from the user is only possible by showing a modal dialog, meaning the simulation of the model is stopped while the user is expected to input information. The animation package presented in this paper avoids this by using an asynchronous event queue polled by a transition in the model. As part of future work, we plan to eliminate polling by allowing external applications to directly produce and consume tokens on special *external places*.

Visualisation is also available in other tool sets. ExSpect [50] allows the user to view the model state by associating widgets with the state of the model and asynchronously interact with the model using simple widgets. In this way, one creates simple user interfaces for displaying information and simple interaction. LTSA [116] allows users to animate models using an animation library called SceneBeans [117]. In LTSA animations are tied to the models by associating each animation activity with a clock; resetting a clock corresponds to starting an animation sequence, and events in the animation corresponds to progress of the clock. PNVis [99] associates objects of a 3D world with tokens, and is suitable for modelling physical systems, but not immediately applicable for network protocols. The Play-Engine [66] supports the developer in implementing a prototype by inputting scenarios (play-in) via an application-specific

GUI, and then execute the resulting program (play-out). Compared to our approach this makes the model implicit as the model is created indirectly via the input scenarios. We view an explicitly created model as an advantage when the prototype is to serve as a basis for an actual implementation of the system. The reason is that an implicitly created model is difficult to interpret as it is automatically generated.
In conclusion, the work presented in this paper has demonstrated that using CP-nets and the supporting computer tools for building a model-based prototype can be a viable and useful alternative to building a physical prototype. Furthermore, the CPN model can also serve as a basis for further development of the interoperability protocol, e.g., by refining the modelling of the routing and dissemination mechanisms to the concrete protocols that would be required to implement the solution. There is still a gap from the CPN model to the actual implementation of the interoperability protocol, but the CPN modelling has yielded an executable prototype that can be used to explore the solution and serve as a basis for the later implementation.

Acknowledgements. The authors gratefully acknowledge the support of their colleagues in BAE SYSTEMS plc, Ericsson Microwave Systems AB and Ericsson Danmark A/S, Telebit, and support from the UK, Swedish and Danish MoDs under the EUCLID/Eurofinder programme, Project RTP6.22 (B2NCW). The authors would also like to acknowledge Rolf Christensen for his contributions.