Routing Protocols in Mobile Ad-hoc Networks

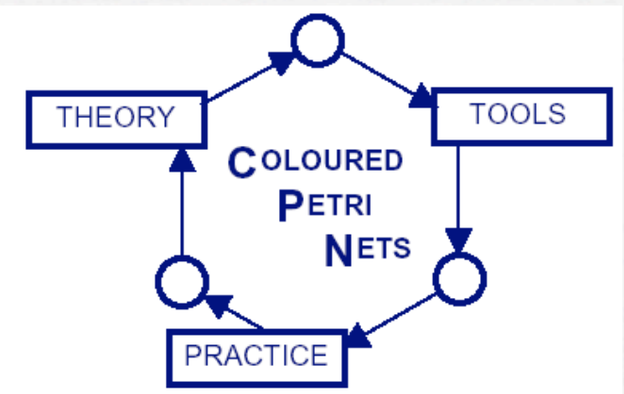
Michael Westergaard
mw@daimi.au.dk
Department of Computer Science
University of Aarhus, Denmark

Overview

- A project on routing in mobile ad-hoc networks
- Modules in coloured Petri nets
- Edge Router Discovery Protocol
- Routing Interoperability Protocol

A project on routing in mobile ad-hoc networks

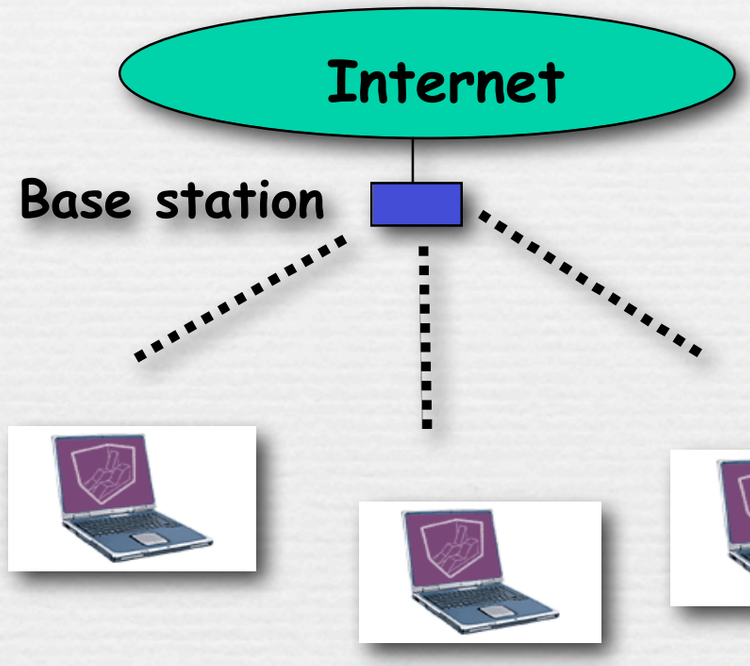
The Project



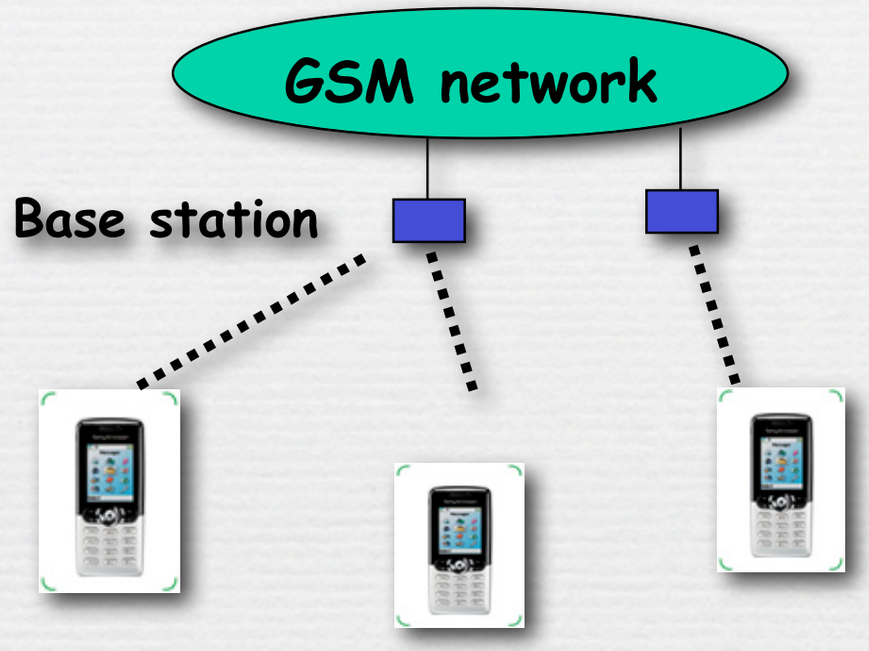
- Participants: Ericsson Denmark A/S, Telebit and CPN Group at University of Aarhus
- Project duration: July 2003-December 2005
- Project web-page: http://www.daimi.au.dk/CPnets/IPv6/
- Executive summary summary: *This project deals with the design and validation of routing protocols and other protocols in ad-hoc and mobile networks*
- The goal was to explore the use of IPv6 in the context of ad-hoc networks using CP-nets

Wireless Communication

W-LAN (e.g. 802.11a/b/g)



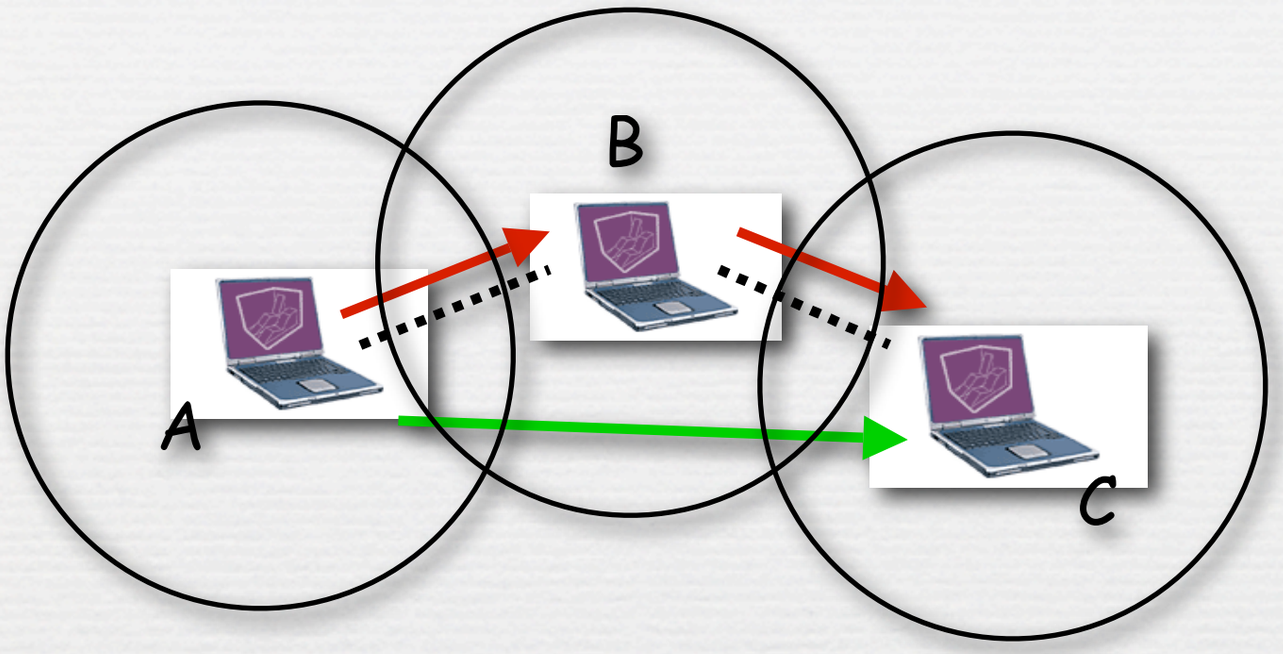
Cellular networks



Key characteristics:

- Communication is based on pre-existing (fixed) infrastructure
- No direct communication between mobile nodes

Mobile Ad-hoc Networks



No pre-existing infrastructure and multi-hop communication

Application areas

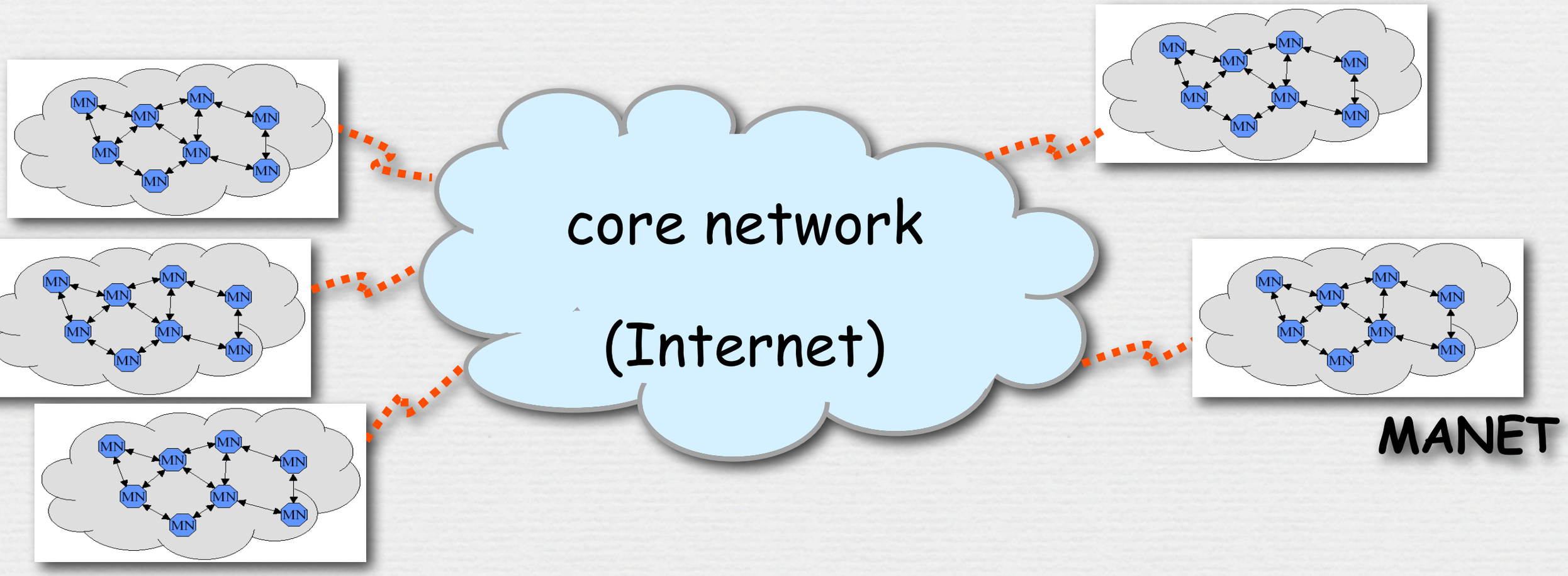
- Sensor networks
- Search-and-rescue operations
- Home networking
- Traffic Safety

Challenges

- Mobility and bandwidth
- Power consumption
- Security
- Fully distributed operation

Hybrid Network Architecture

A main topic of the project was protocols for integration of fixed core networks and mobile ad-hoc networks



Sub-projects

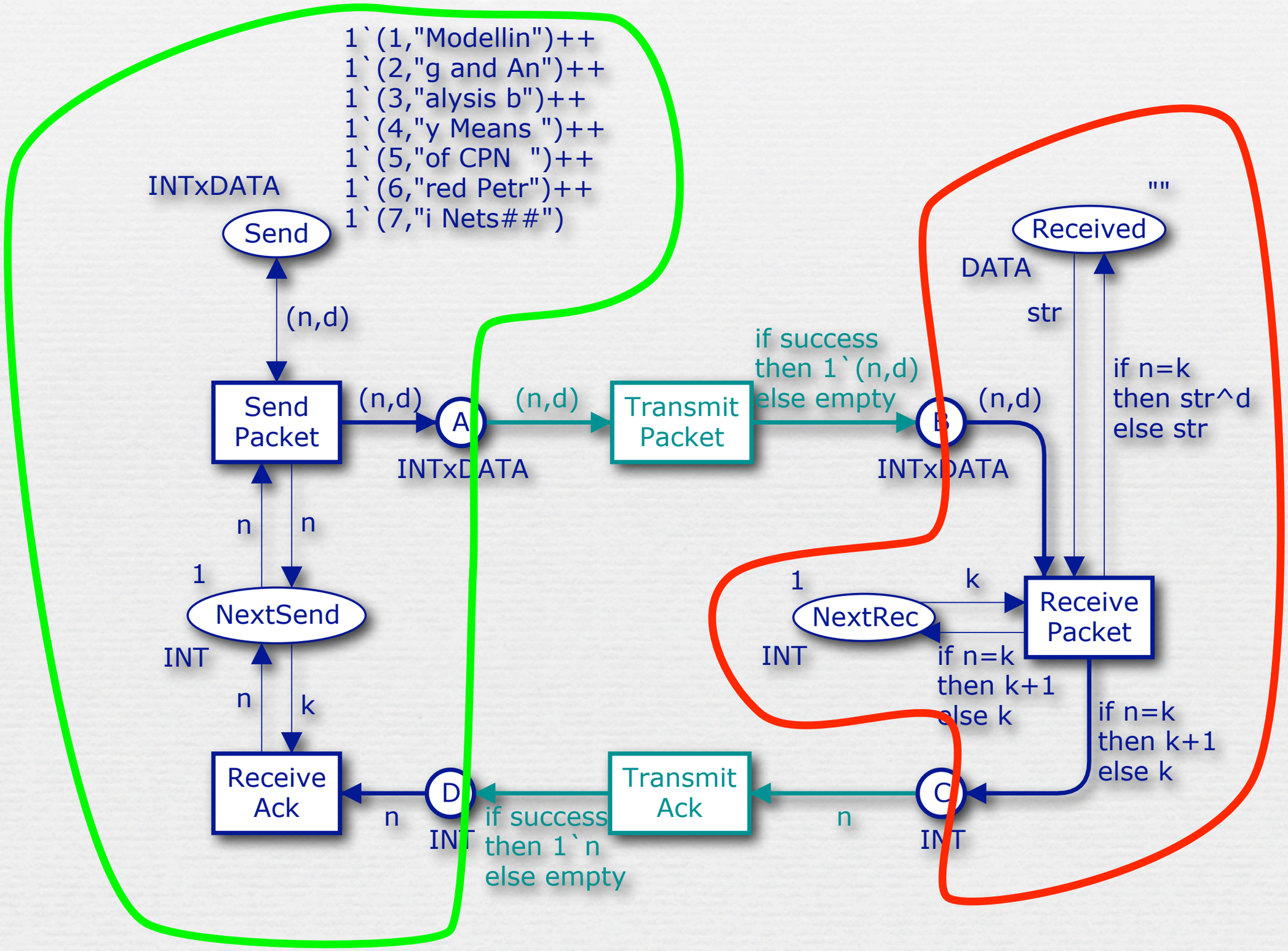
1) Specification of mobility and communication scenarios in an Internet-MANET network architecture

2) Specification of an Edge Router Discovery Protocol for mobile ad-hoc networks

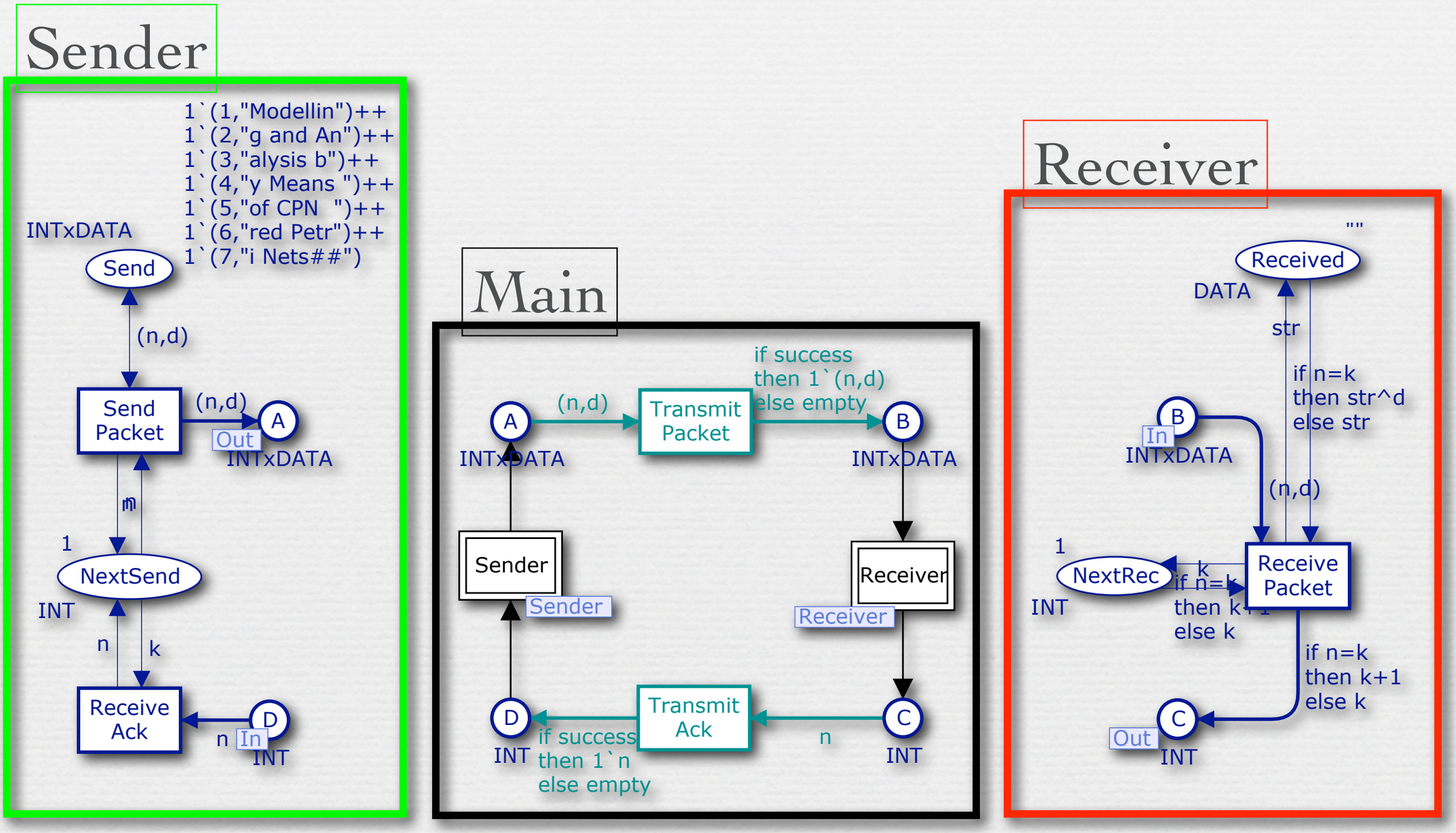
3) Model-based prototyping of protocols for Internet-MANET routing with redundant gateways

Modules in coloured Petri nets

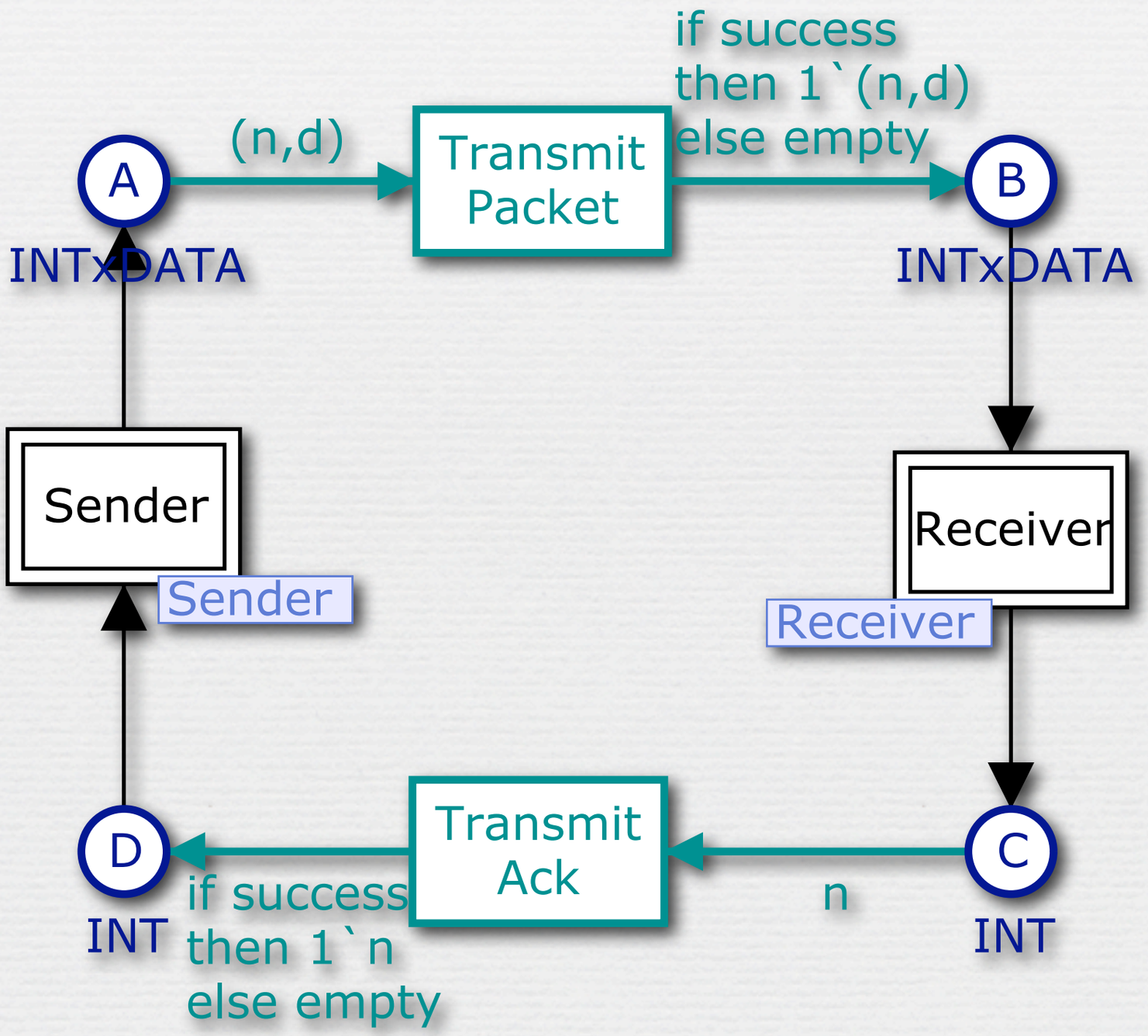
A CPN Model


1`(1,"Modellin")++
1`(2,"g and An")++
1`(3,"alysis b")++
1`(4,"y Means ")++
1`(5,"of CPN ")++
1`(6,"red Petr")++
1`(7,"i Nets##")
INTxDATA
Send
(n,d)
Send Packet
(n,d)
A
INTxDATA
(n,d)
Transmit Packet
if success
then 1`(n,d)
else empty
B
INTxDATA
(n,d)
""
Received
DATA
str
if n=k
then str^d
else str
n
n
1
NextSend
INT
n
k
1
k
NextRec
INT
if n=k
then k+1
else k
Receive Packet
if n=k
then k+1
else k
Receive Ack
n
D
INT
if success
then 1`n
else empty
Transmit Ack
n
C
INT

A Simpler CPN model



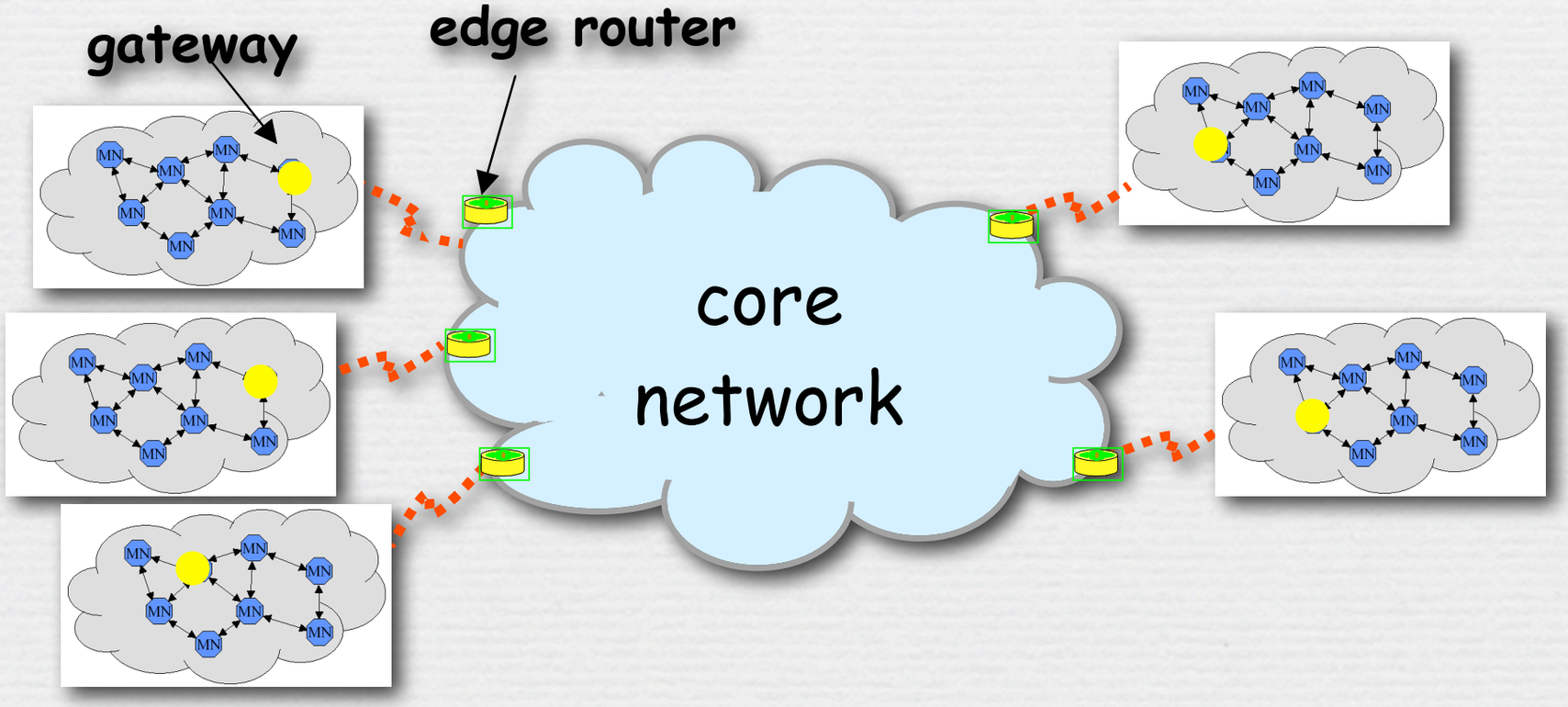
Main Module


if success
then 1`(n,d)
else empty
(n,d)
Transmit
Packet
A
B
INTxDATA
INTxDATA
Sender
Sender
Receiver
Receiver
Transmit
Ack
D
C
if success
then 1`n
else empty
n
INT
INT

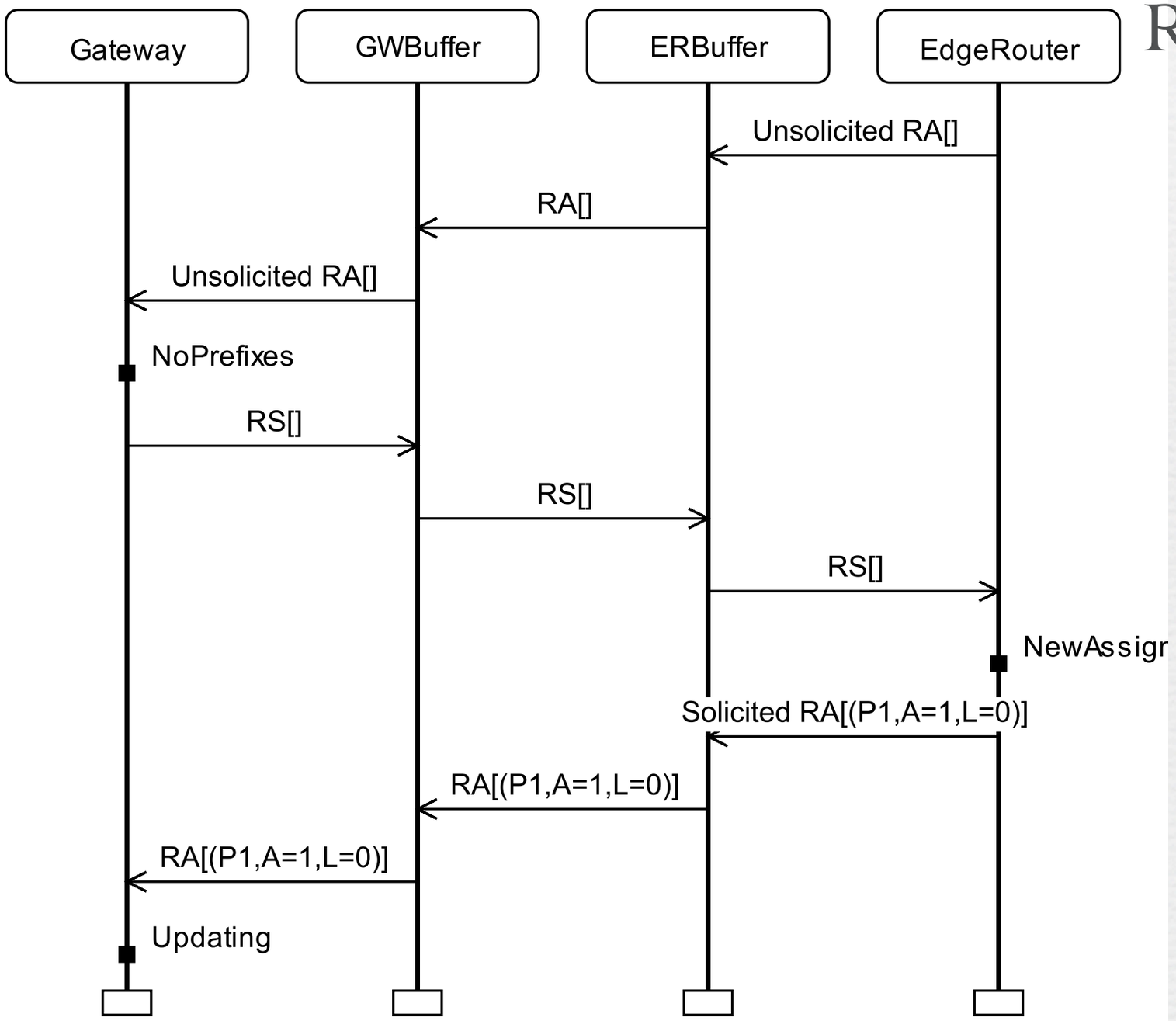
Edge Router Discovery Protocol

Edge Router Discovery Protocol (ERDP)

ERDP allows edge routers to configure gateways with address prefixes



Basic Operation of ERDP



RA: Router Advertisement
RS: Router Solicitation

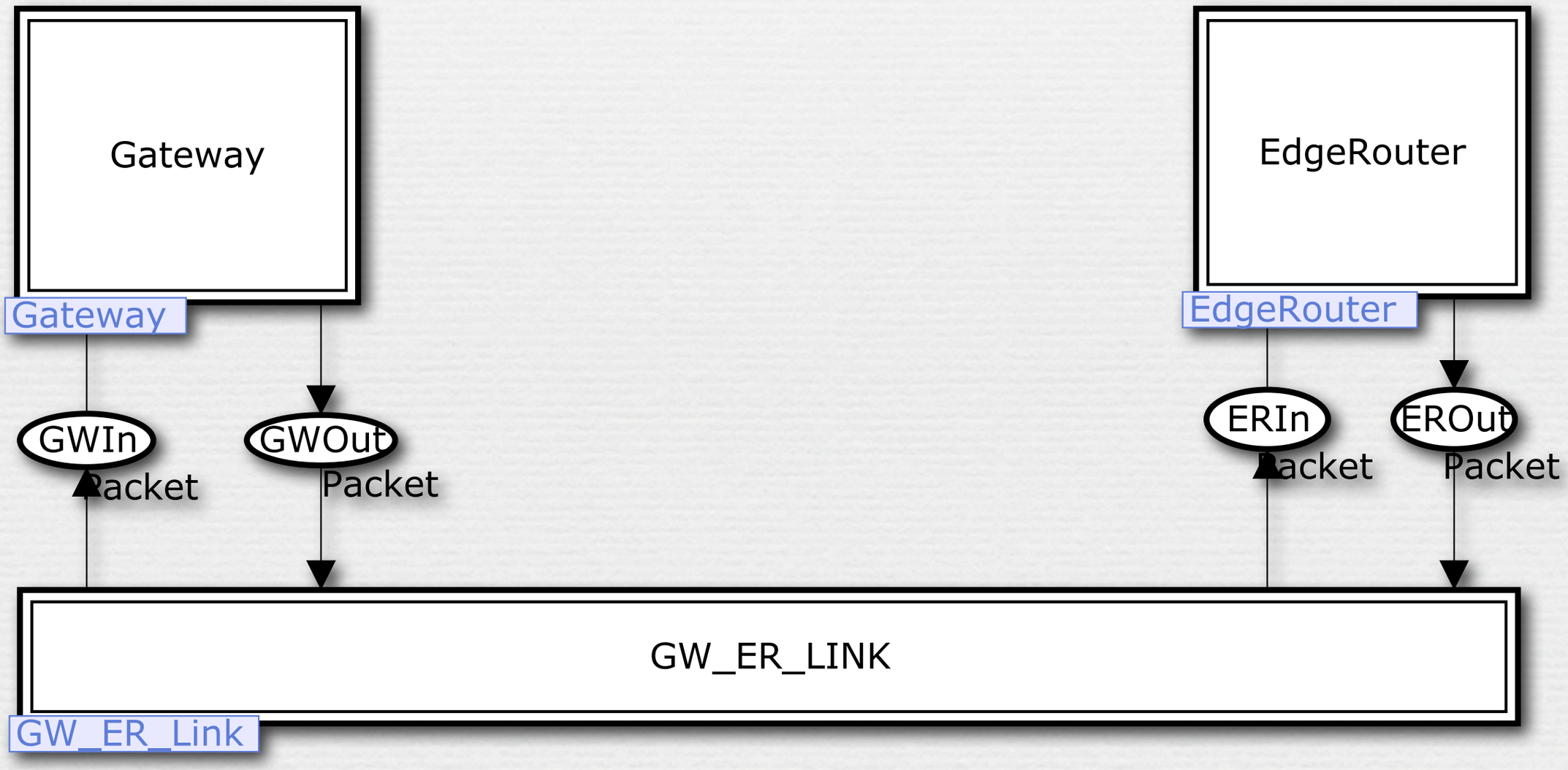
ERDP requirements

- mobility of gateways
- expire of address prefixes
- unreliable wireless links

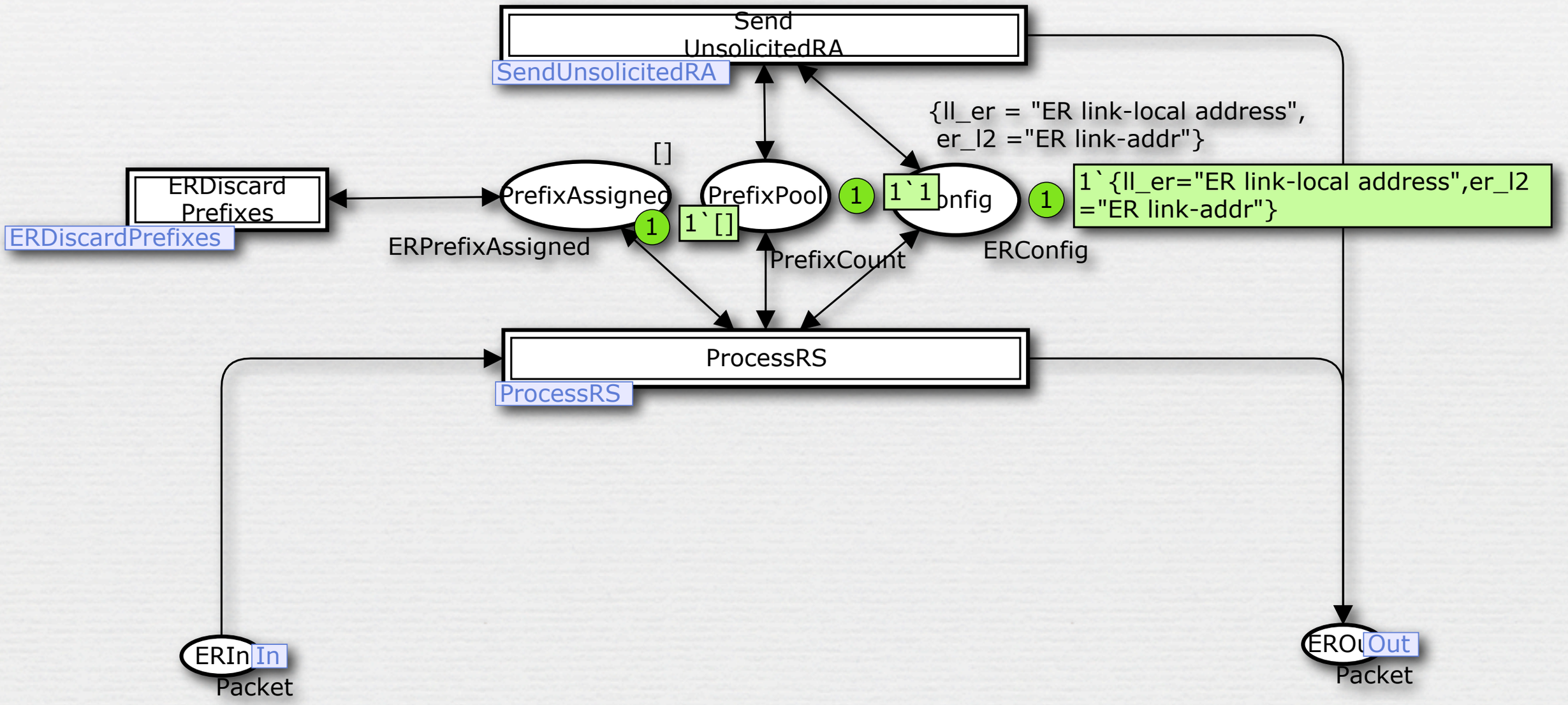
Modelling Phase

- Natural language specification developed by protocol engineers from Ericsson Denmark A/S, Telebit
- CPN model reflecting the specification developed by researchers from the CPN group
- Protocol developers were given a 6 hour course enabling them to read and interpret CPN models
- Approximately 70 man-hours were used on modelling

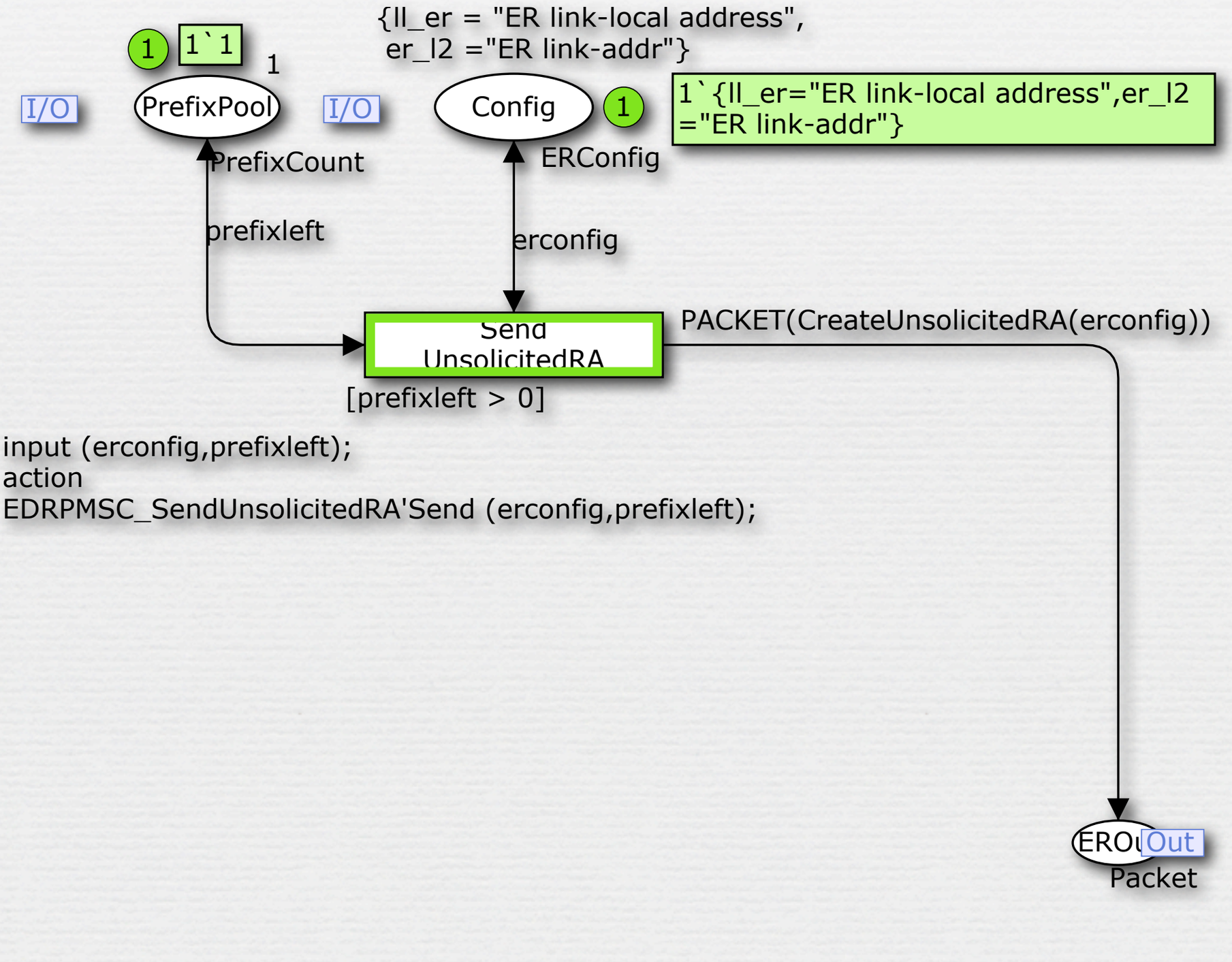
ERDP


Gateway
Gateway
EdgeRouter
EdgeRouter
GWIn
Packet
GWOut
Packet
ERIn
Packet
EROut
Packet
GW_ER_LINK
GW_ER_Link

Edge Router


Send
UnsolicitedRA
SendUnsolicitedRA
{ll_er = "ER link-local address",
er_l2 ="ER link-addr"}
[]
ERDiscard
Prefixes
ERDiscardPrefixes
PrefixAssigned
1
ERPrefixAssigned
PrefixPool
1`[]
1
1`1
onfig
1
ERConfig
1`{ll_er="ER link-local address",er_l2
="ER link-addr"}
PrefixCount
ProcessRS
ProcessRS
ERIn
In
Packet
ERO
Out
Packet

Send Unsolicited RA



Results from Modelling

- Several design issues were identified during modelling

Category	Review 1	Review 2	Total
Imcompleteness and ambiguity in specification	3	6	9
Errors in protocol	2	7	9
Simplifications of protocol	2	0	2
Additions	4	0	4
Total	11	13	24

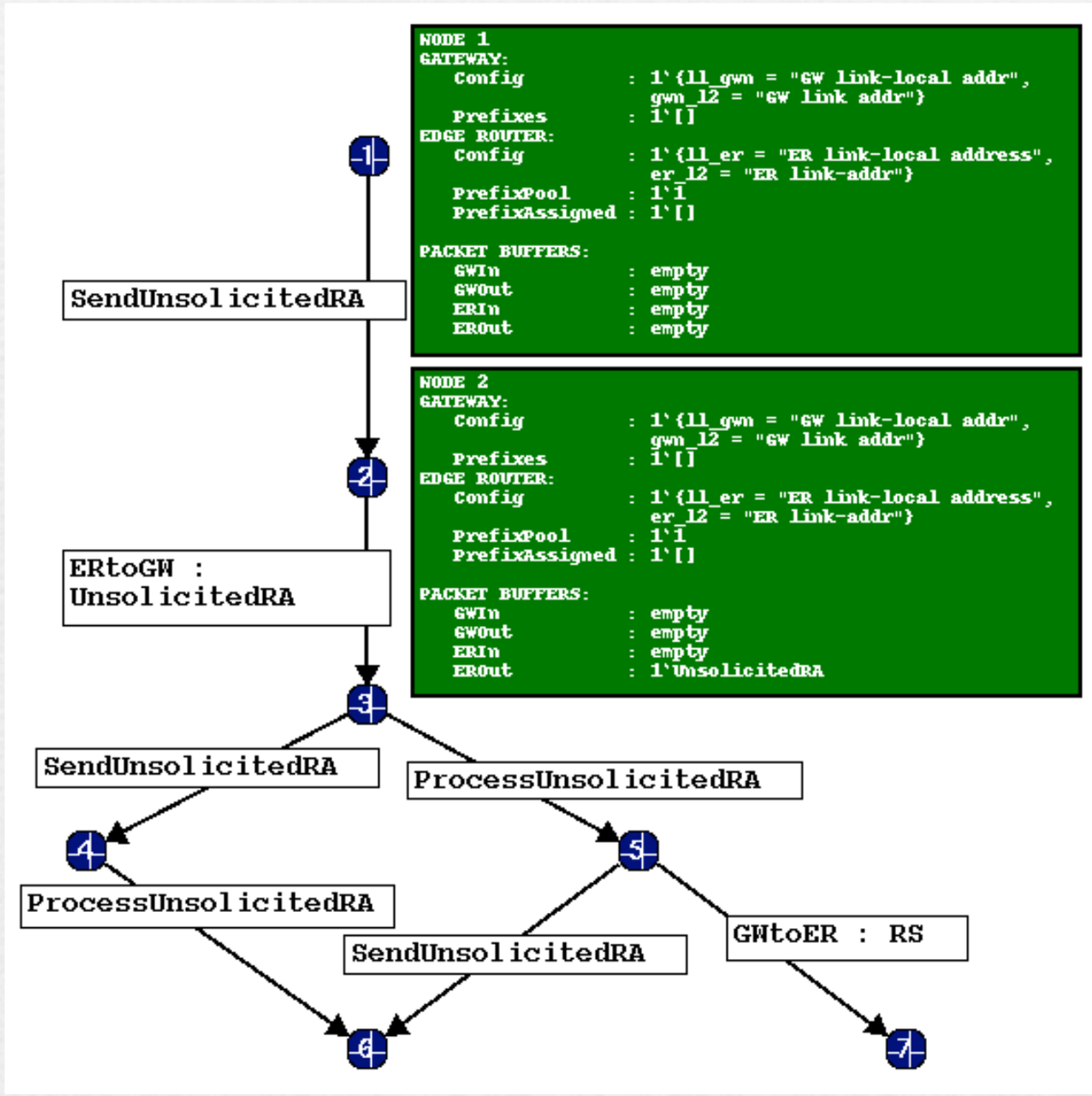
State Space Analysis

Nodes = reachable states

Arcs = actions

Paths = executions

- Highly **automatic**
- Counter-examples



Analysis Approach

- **Key property**: From any state with a non-configured prefix, it is possible to reach a state where the prefix is consistently configured
- Analysis in 3 steps
  i. Basic configuration
  ii. Packet loss allowed
  iii. Expire of prefixes allowed

Analysis Results

- Basic configuration
  - Synchronisation error between edge router and gateway
- Packet loss allowed
  - Synchronisation error when certain unsolicited RAs were lost
  - Error in processing of RA in gateway (livelock)
- Expire of prefixed allowed
  - No additional errors

Conclusions

- The act of constructing a CPN model provided **valuable input** to the ERDP specification
- Simulation and graphical feedback using message sequence charts **added insight** into the operation of the protocol
- State-space **analysis revealed 3 errors** and the key property of the revised protocol could be verified

Experiences

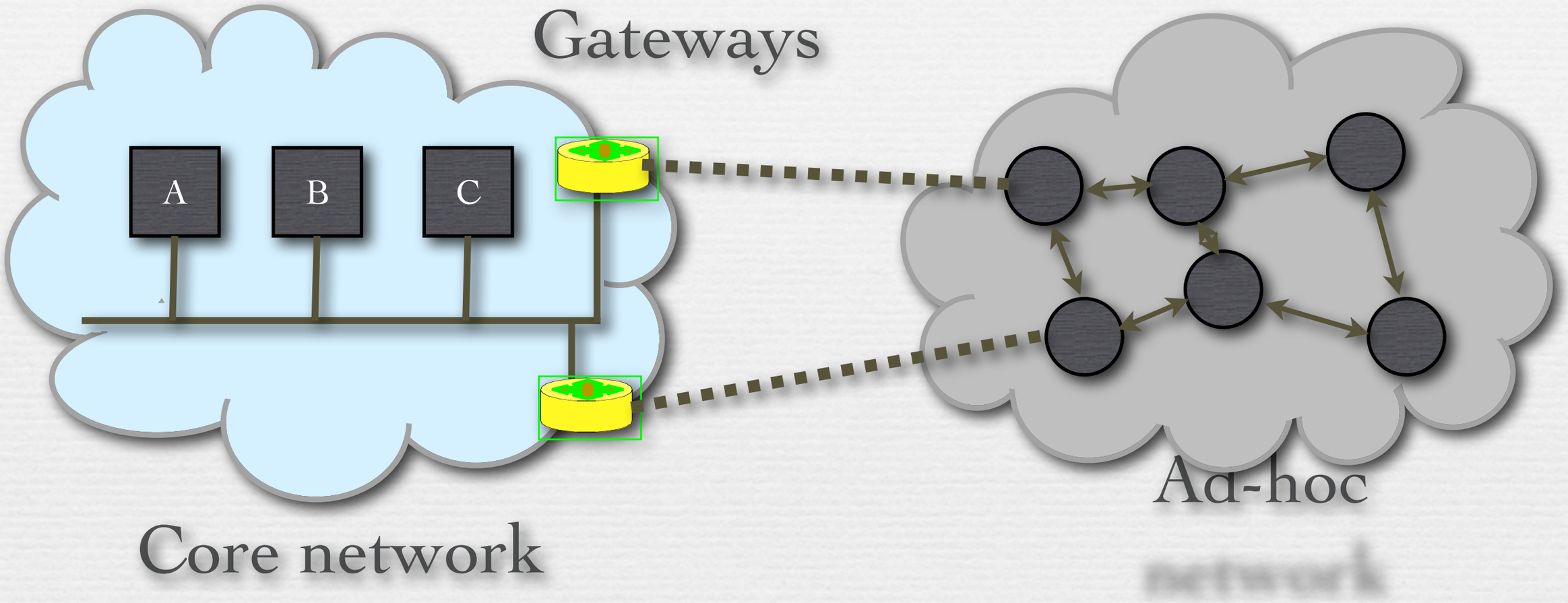
Using CP-nets was a **success**, as

- The modelling language and the supporting tools were **powerful** enough to specify and validate a **real-world** protocol
- Several non-trivial design issues and errors were identified and fixed
- Approximately 100 man-hours over a period of 4 months to create the model and analyse it

Routing Interoperability Protocol

Network Architecture

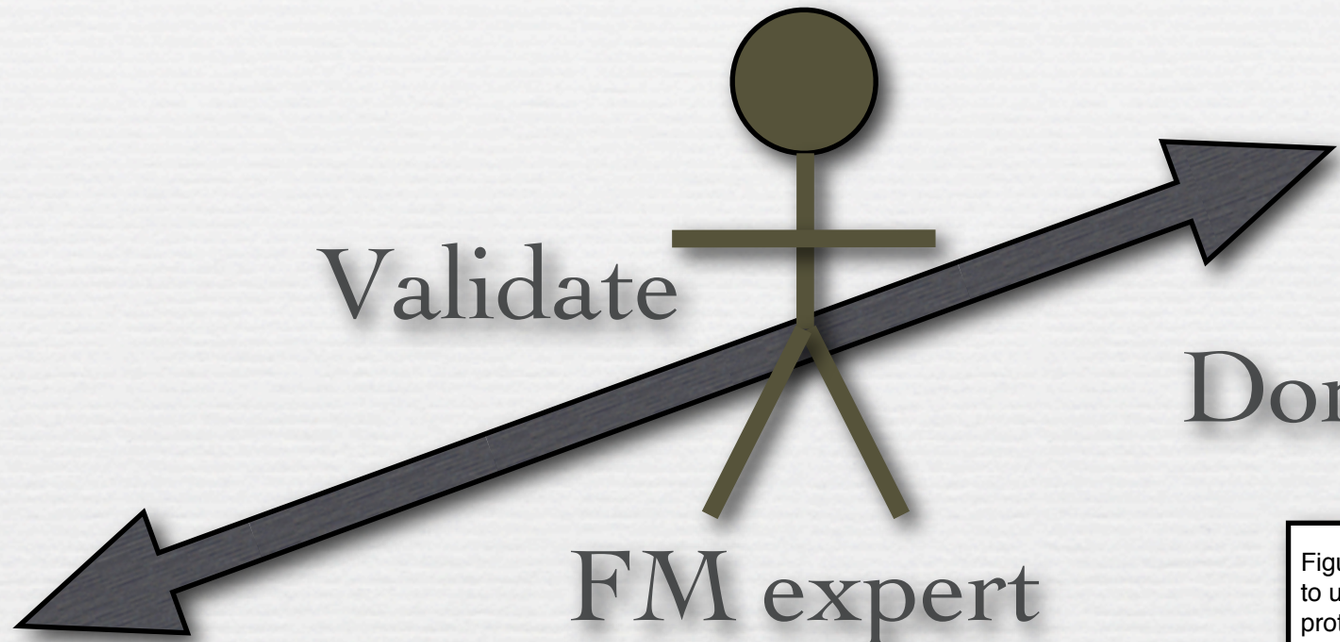
Gateways



Possible solutions

- Mobile IP
- Mobile host routes injected by gateways into the core network
- Dynamic DNS and renumbering

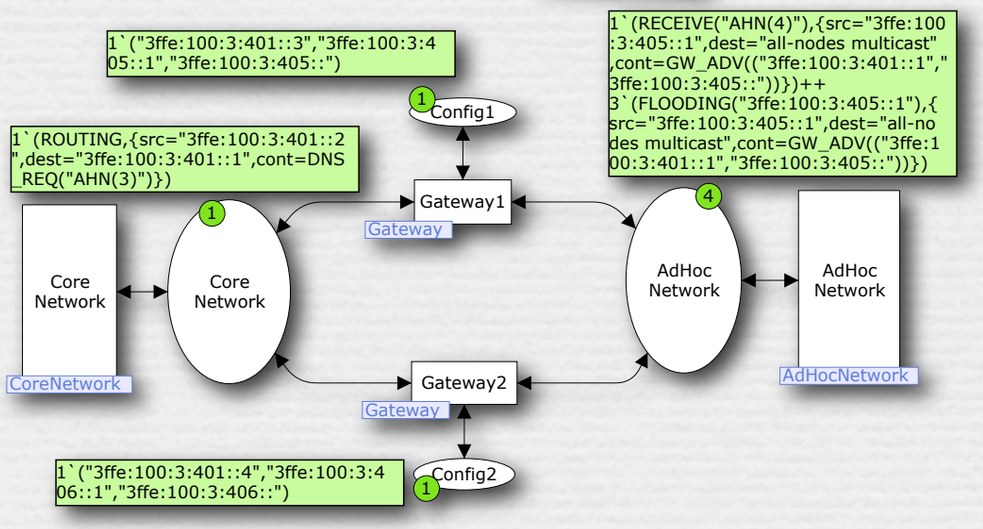
Model-based Prototyping



Domain expert

FM expert

Modeling

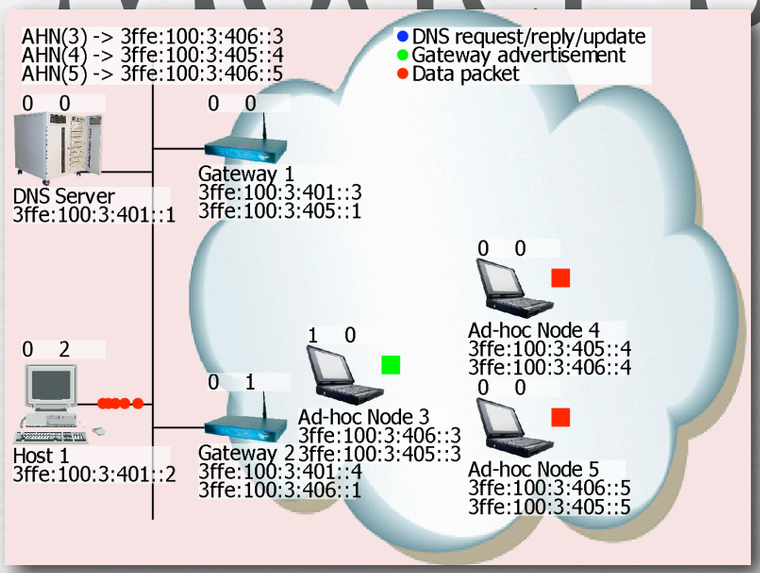


Formal model

Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (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 Fig. 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

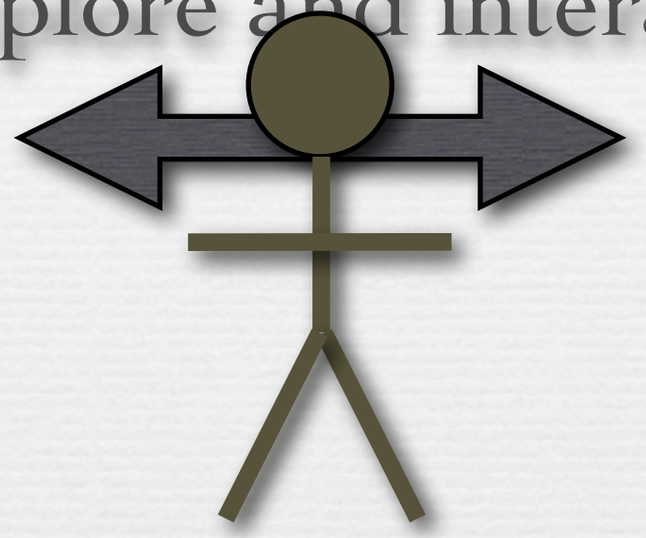
Specification

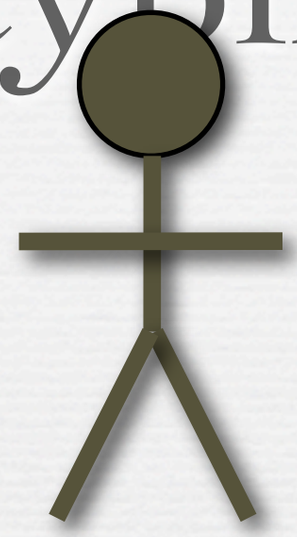
Model-based Prototyping



Animation

Explore and interact

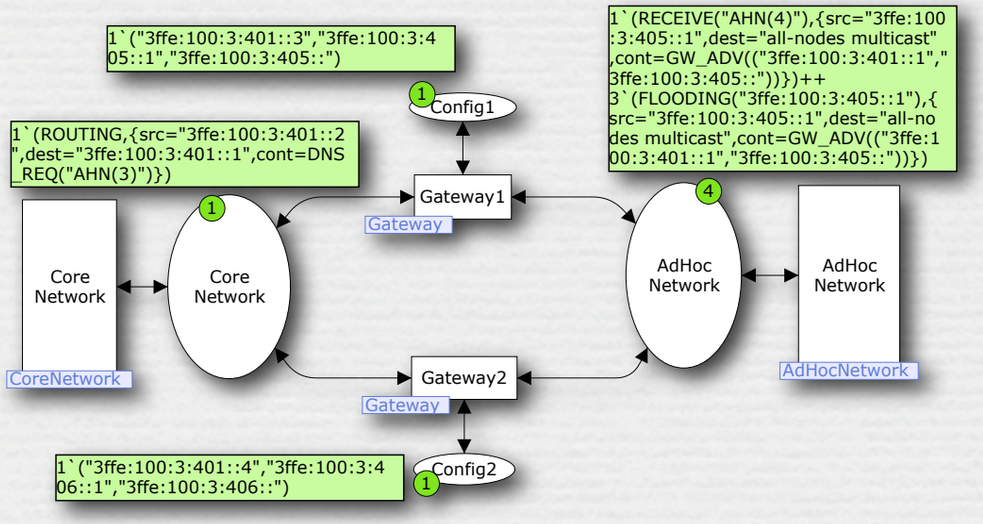




Domain expert

FM expert

Modeling

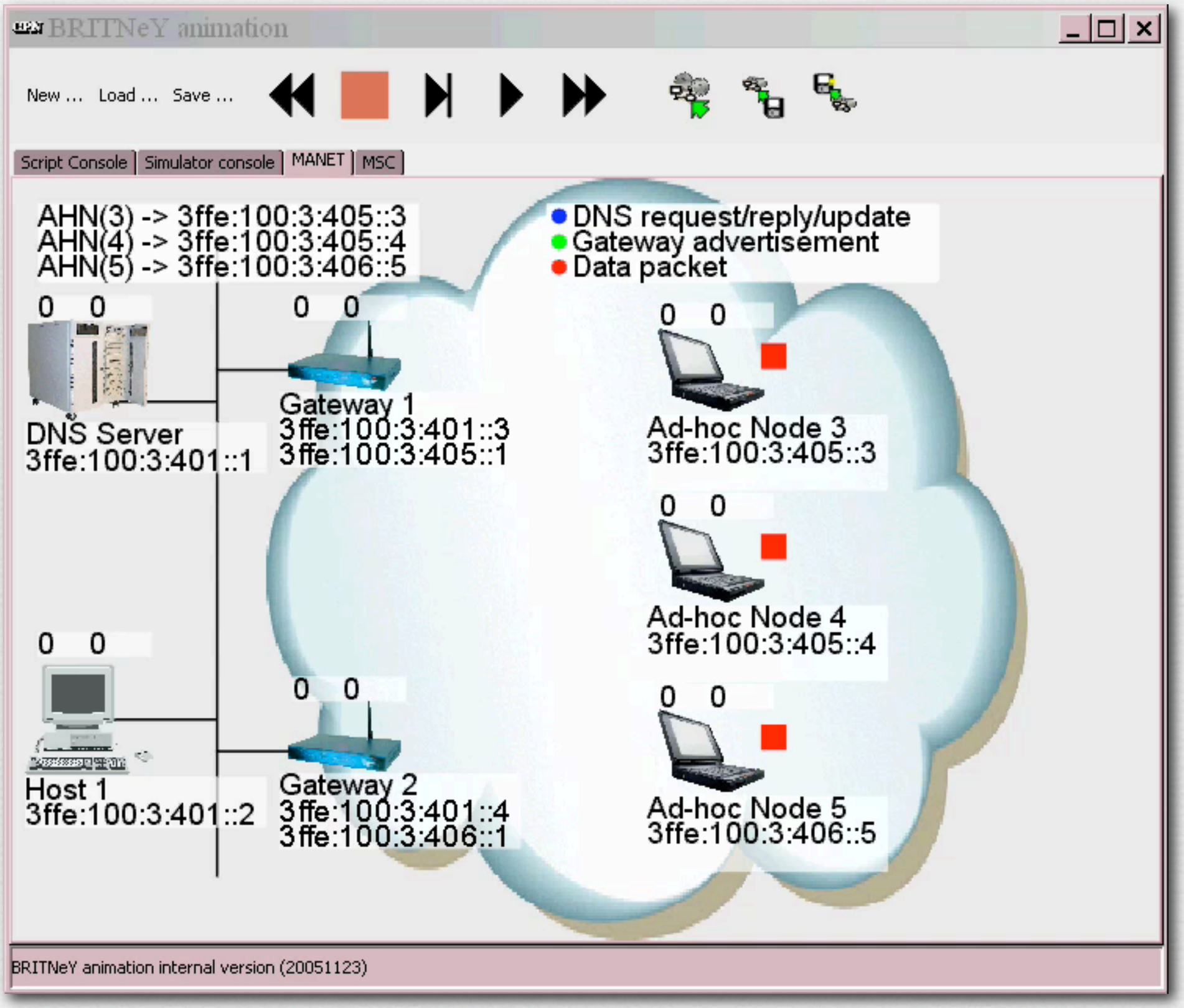


Formal model

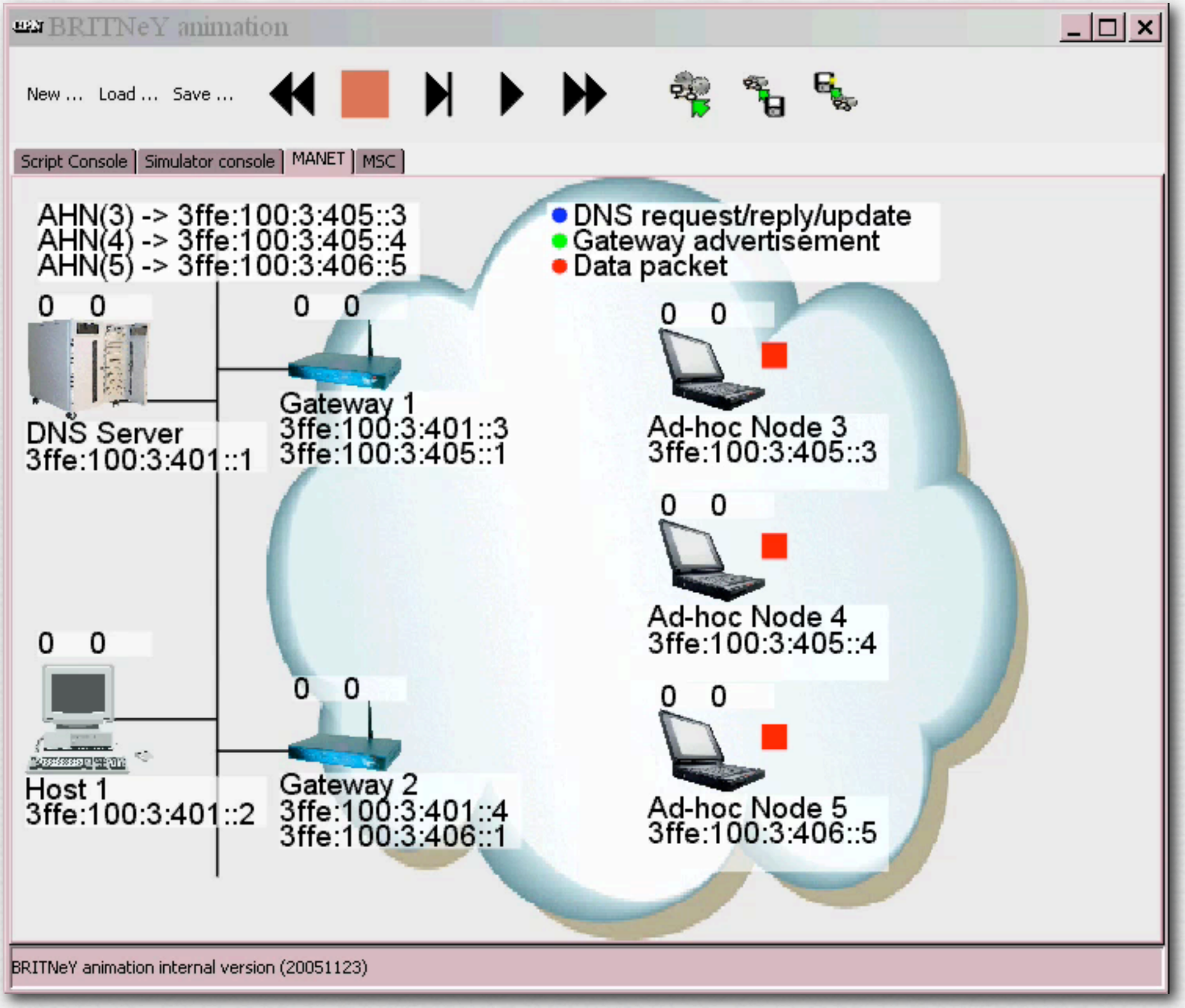
Figure 2 shows the approach taken to use CPN models to develop a prototype of the interoperability protocol. A CPN model (lower left of Fig. 2) has been developed by modelling the natural language protocol specification [22] (lower right) of the interoperability protocol. The modelling activity transforms the natural language specification into a formal executable specification represented by the CPN model. The CPN model captures the network architecture depicted in Fig. 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

Specification

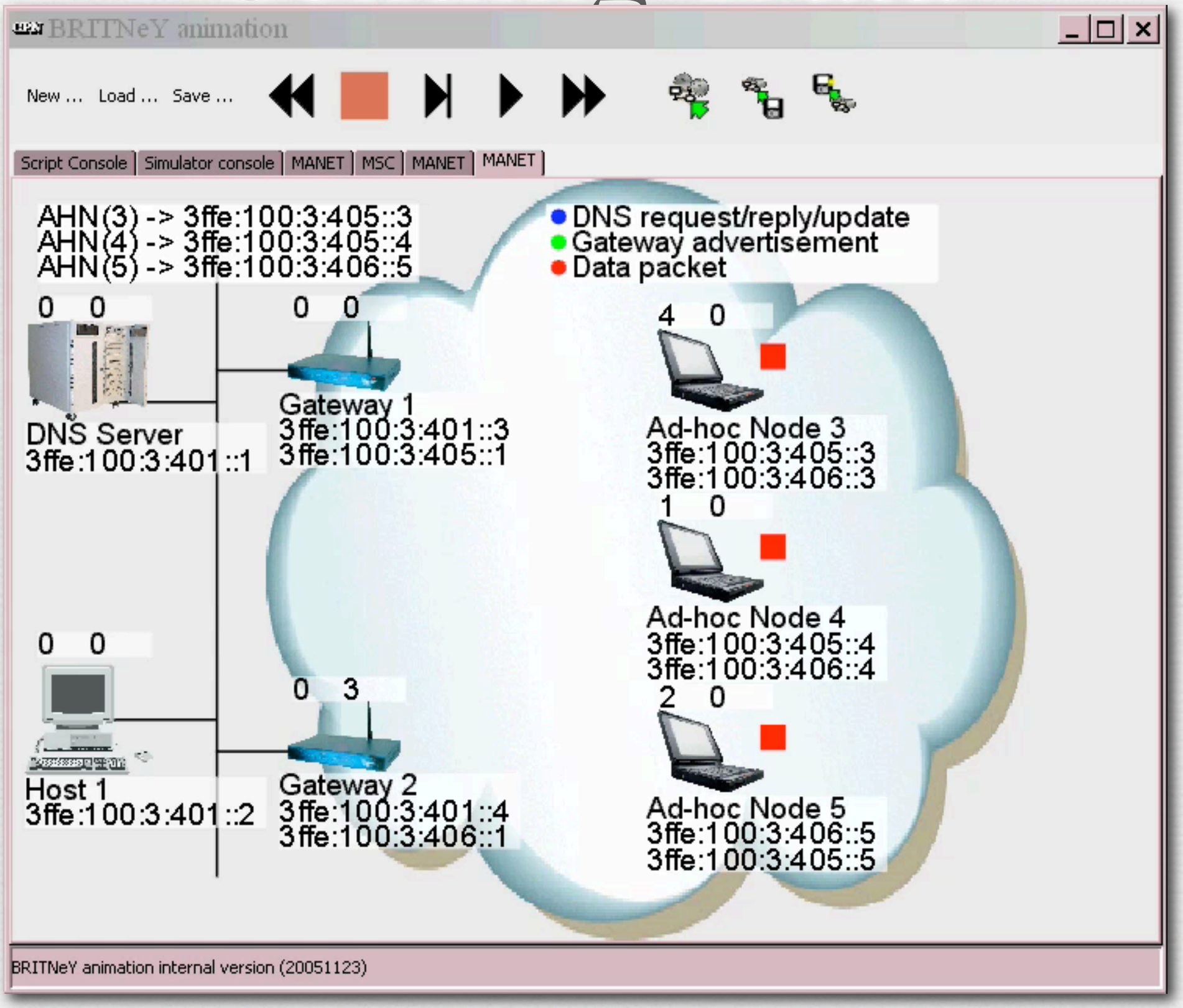
Scenario


BRITNeY animation
New ... Load ... Save ...
Script Console
Simulator console
MANET
MSC
AHN(3) -> 3ffe:100:3:405::3
AHN(4) -> 3ffe:100:3:405::4
AHN(5) -> 3ffe:100:3:406::5
DNS request/reply/update
Gateway advertisement
Data packet
DNS Server
3ffe:100:3:401::1
Gateway 1
3ffe:100:3:401::3
3ffe:100:3:405::1
Ad-hoc Node 3
3ffe:100:3:405::3
Ad-hoc Node 4
3ffe:100:3:405::4
Host 1
3ffe:100:3:401::2
Gateway 2
3ffe:100:3:401::4
3ffe:100:3:406::1
Ad-hoc Node 5
3ffe:100:3:406::5
BRITNeY animation internal version (20051123)

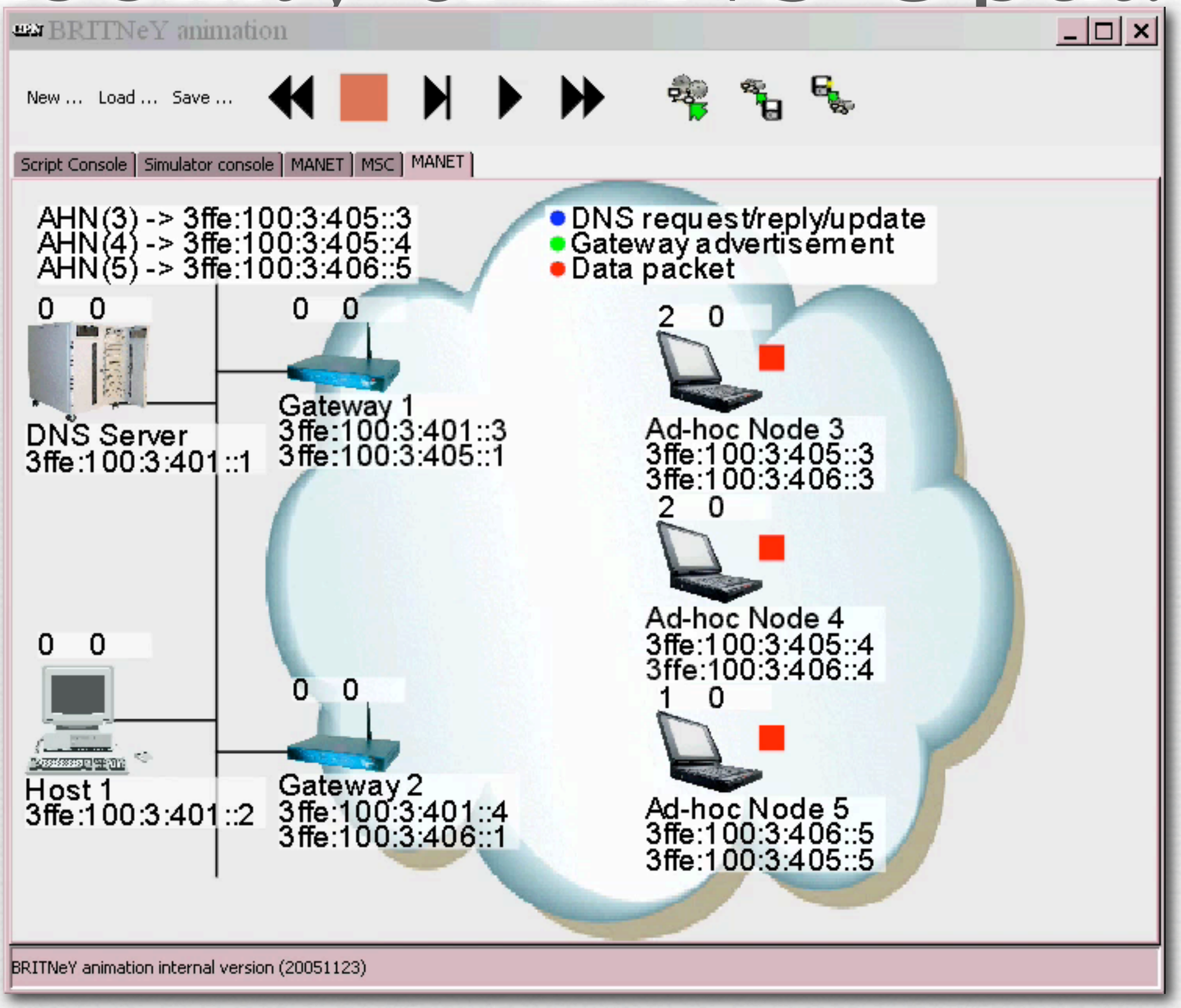
Router Advertisements



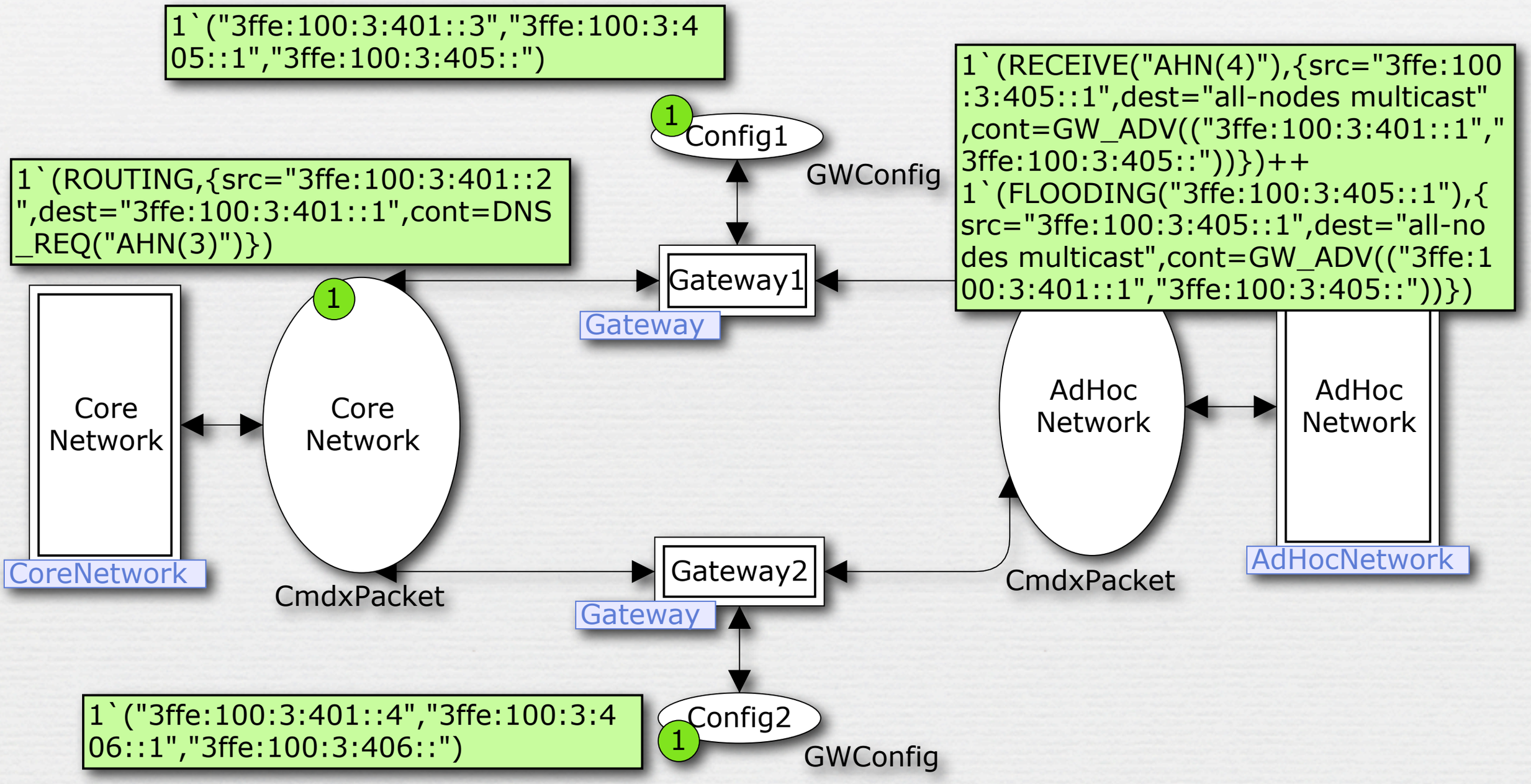
Sending Data



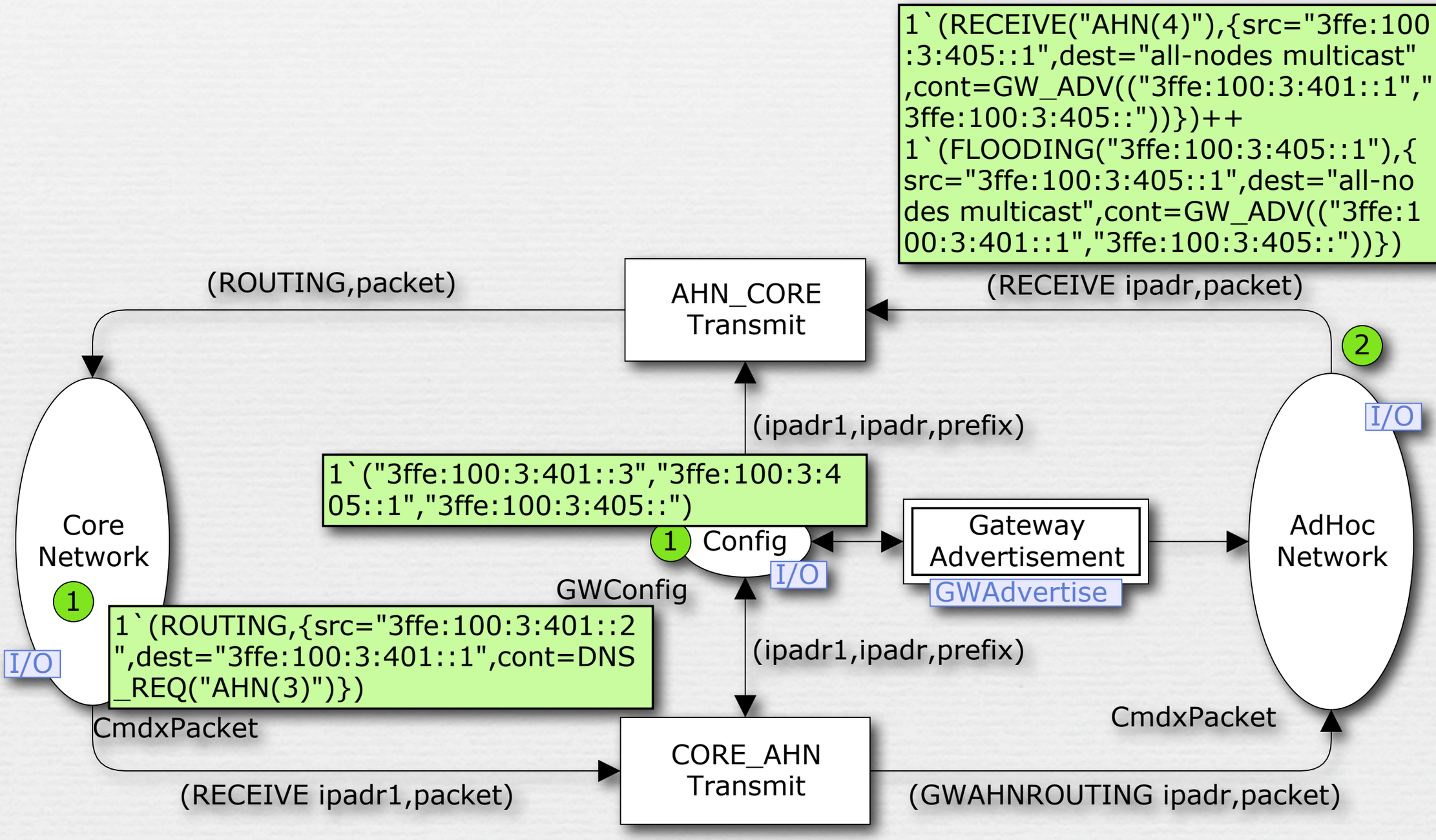
Mobility & DNS Update



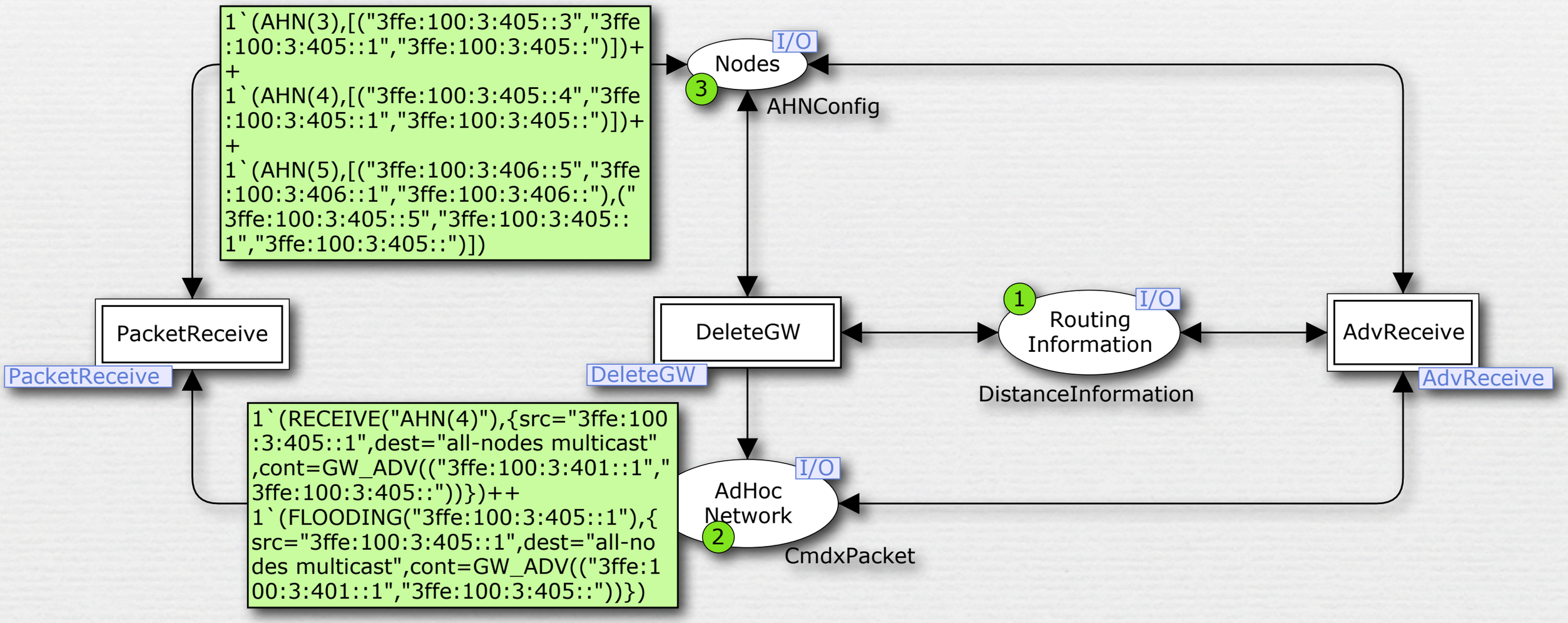
Model


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

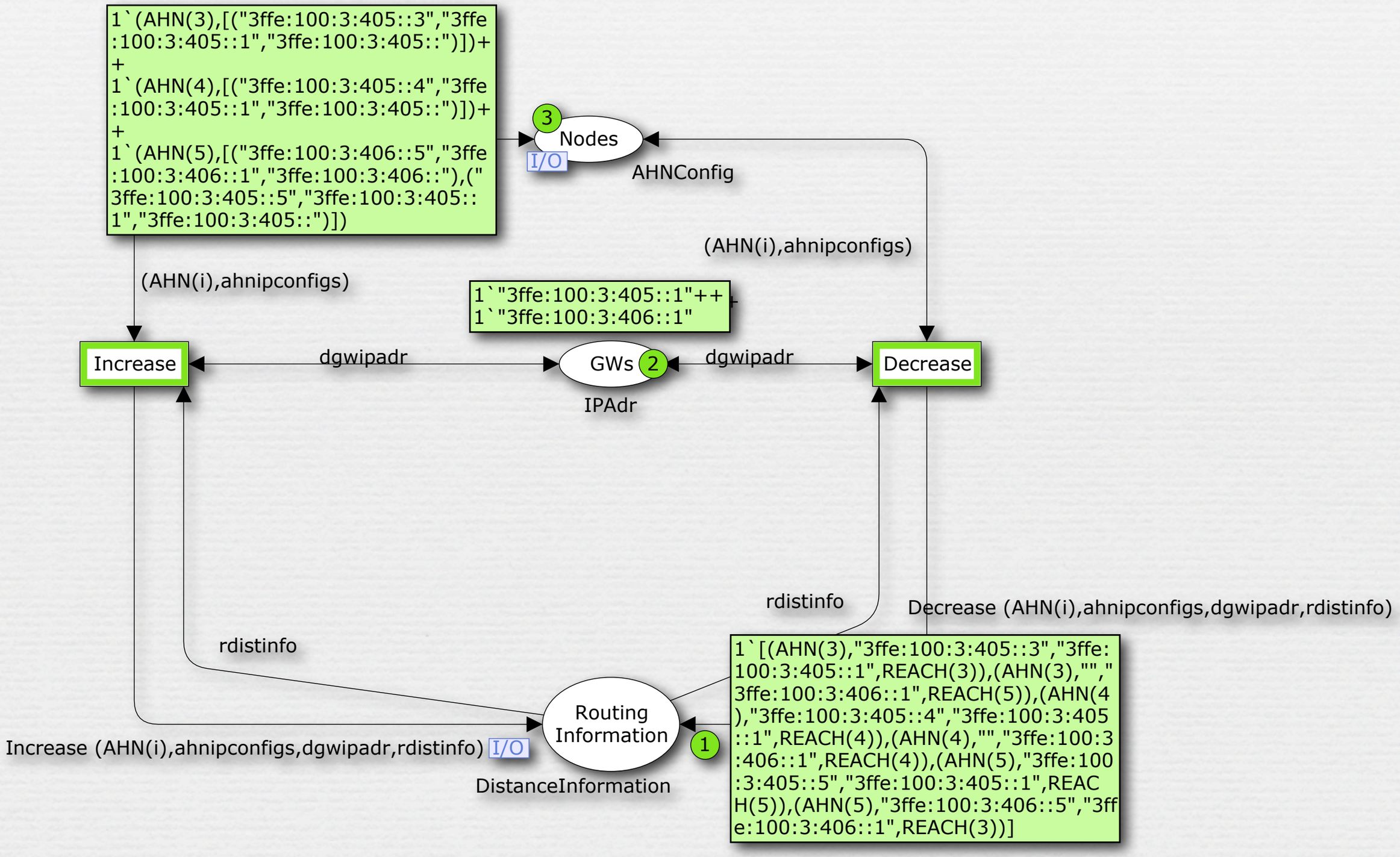
Gateway


1`(RECEIVE("AHN(4)"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})++
1`(FLOODING("3ffe:100:3:405::1"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})
(ROUTING,packet)
AHN_CORE Transmit
(RECEIVE ipadr,packet)
2
I/O
(ipadr1,ipadr,prefix)
1`("3ffe:100:3:401::3","3ffe:100:3:405::1","3ffe:100:3:405::")
Core Network
1
Config
I/O
Gateway Advertisement
GWAdvertise
AdHoc Network
GWConfig
1`(ROUTING,{src="3ffe:100:3:401::2",dest="3ffe:100:3:401::1",cont=DNS_REQ("AHN(3)")})
I/O
(ipadr1,ipadr,prefix)
CmdxPacket
CmdxPacket
CORE_AHN Transmit
(RECEIVE ipadr1,packet)
(GWAHNROUTING ipadr,packet)

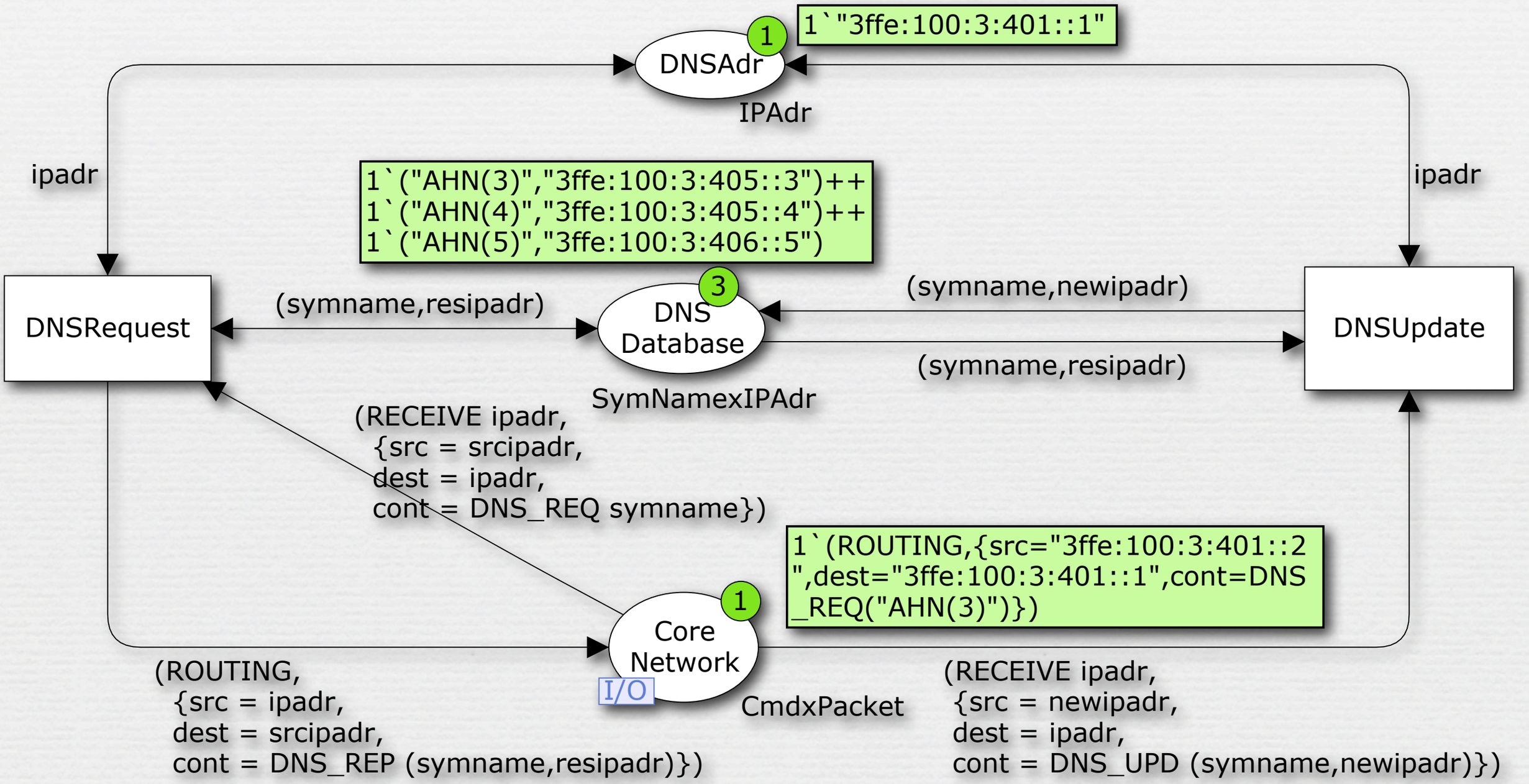
Ad-hoc Node


1`(AHN(3),[("3ffe:100:3:405::3","3ffe:100:3:405::1","3ffe:100:3:405::")])++
1`(AHN(4),[("3ffe:100:3:405::4","3ffe:100:3:405::1","3ffe:100:3:405::")])++
1`(AHN(5),[("3ffe:100:3:406::5","3ffe:100:3:406::1","3ffe:100:3:406::"),("3ffe:100:3:405::5","3ffe:100:3:405::1","3ffe:100:3:405::")])
Nodes
I/O
3
AHNConfig
PacketReceive
PacketReceive
DeleteGW
DeleteGW
1
Routing Information
I/O
DistanceInformation
AdvReceive
AdvReceive
1`(RECEIVE("AHN(4)"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})++
1`(FLOODING("3ffe:100:3:405::1"),{src="3ffe:100:3:405::1",dest="all-nodes multicast",cont=GW_ADV(("3ffe:100:3:401::1","3ffe:100:3:405::"))})
AdHoc Network
I/O
2
CmdxPacket

Mobility


1`(AHN(3),[("3ffe:100:3:405::3","3ffe:100:3:405::1","3ffe:100:3:405::")])++
1`(AHN(4),[("3ffe:100:3:405::4","3ffe:100:3:405::1","3ffe:100:3:405::")])++
1`(AHN(5),[("3ffe:100:3:406::5","3ffe:100:3:406::1","3ffe:100:3:406::"),("3ffe:100:3:405::5","3ffe:100:3:405::1","3ffe:100:3:405::")])
3
Nodes
I/O
AHNConfig
(AHN(i),ahnipconfigs)
(AHN(i),ahnipconfigs)
1`"3ffe:100:3:405::1"++
1`"3ffe:100:3:406::1"
Increase
dgwipadr
GWs
2
IPAdr
dgwipadr
Decrease
rdistinfo
Decrease (AHN(i),ahnipconfigs,dgwipadr,rdistinfo)
rdistinfo
1`[(AHN(3),"3ffe:100:3:405::3","3ffe:100:3:405::1",REACH(3)),(AHN(3),"","3ffe:100:3:406::1",REACH(5)),(AHN(4),"3ffe:100:3:405::4","3ffe:100:3:405::1",REACH(4)),(AHN(4),"","3ffe:100:3:406::1",REACH(4)),(AHN(5),"3ffe:100:3:405::5","3ffe:100:3:405::1",REACH(5)),(AHN(5),"3ffe:100:3:406::5","3ffe:100:3:406::1",REACH(3))]
Routing Information
1
Increase (AHN(i),ahnipconfigs,dgwipadr,rdistinfo)
I/O
DistanceInformation

DNS Server


1`"3ffe:100:3:401::1"
1
DNSAdr
IPAdr
ipadr
ipadr
1`("AHN(3)","3ffe:100:3:405::3")++
1`("AHN(4)","3ffe:100:3:405::4")++
1`("AHN(5)","3ffe:100:3:406::5")
DNSRequest
(symname,resipadr)
3
DNS
Database
SymNamexIPAdr
(symname,newipadr)
DNSUpdate
(symname,resipadr)
(RECEIVE ipadr,
{src = srcipadr,
dest = ipadr,
cont = DNS_REQ symname})
1`(ROUTING,{src="3ffe:100:3:401::2
",dest="3ffe:100:3:401::1",cont=DNS
_REQ("AHN(3)")})
1
Core
Network
I/O
CmdxPacket
(ROUTING,
{src = ipadr,
dest = srcipadr,
cont = DNS_REP (symname,resipadr)})
(RECEIVE ipadr,
{src = newipadr,
dest = ipadr,
cont = DNS_UPD (symname,newipadr)})

Advantages of Model-based Prototypes

- Easier to control and reproduce scenarios
- Implementation details can be abstracted away
- Setup of physical network equipment is not required
- Larger scenarios can be investigated

Advantages of Integration of CP-nets with Animation

- Behaviour is as defined by the **formal** model
- Knowledge of the formal modelling language is **not required**
- Presentation for military leaders is possible
- Validation that the implemented prototype corresponds to the specification