

# Routing Protocols in Mobile Ad-hoc Networks

Michael Westergaard

`mw@daimi.au.dk`

Department of Computer Science  
University of Aarhus

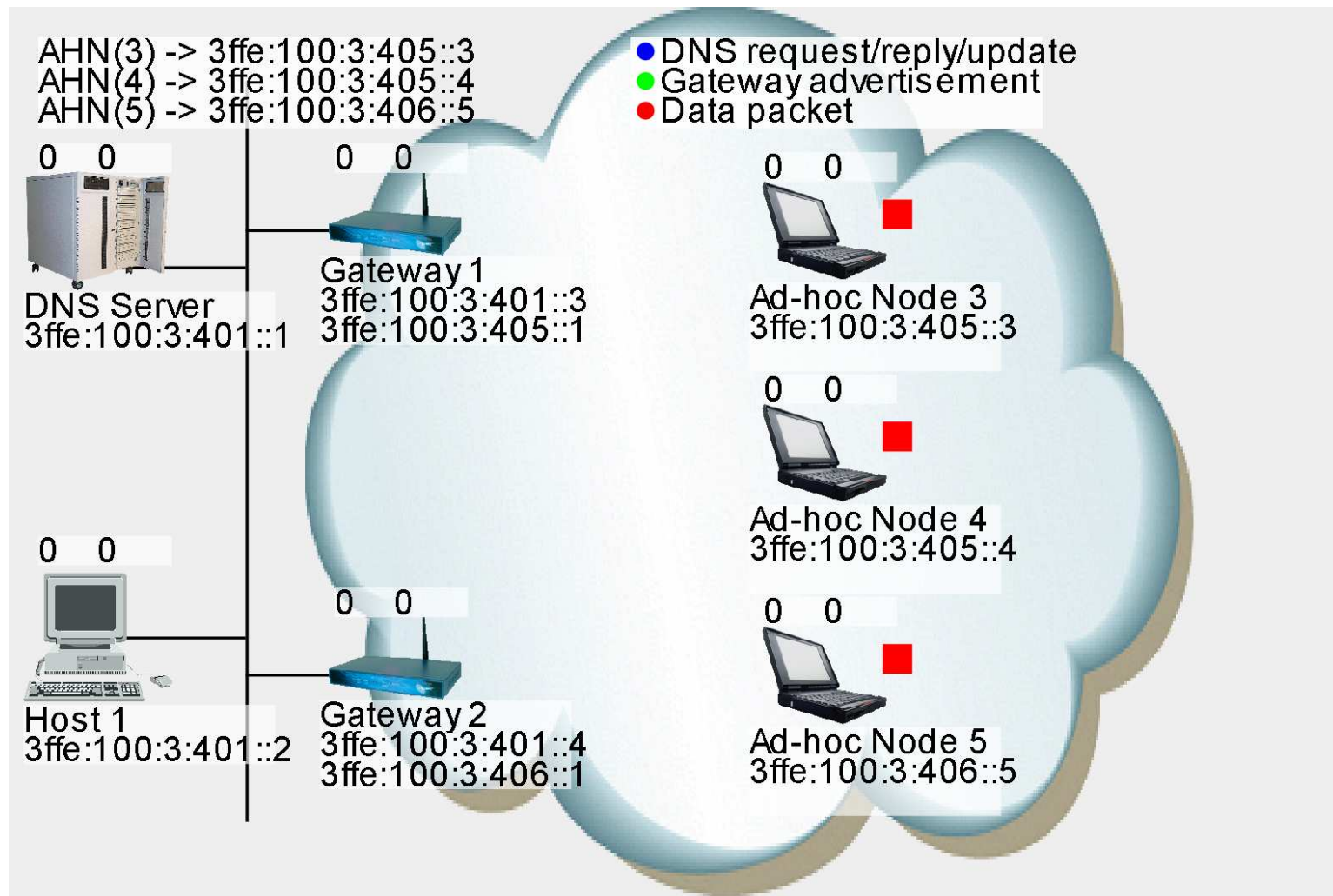
# Overview

- A project on routing in mobile ad-hoc networks
- Modules in coloured Petri-nets
- Demo

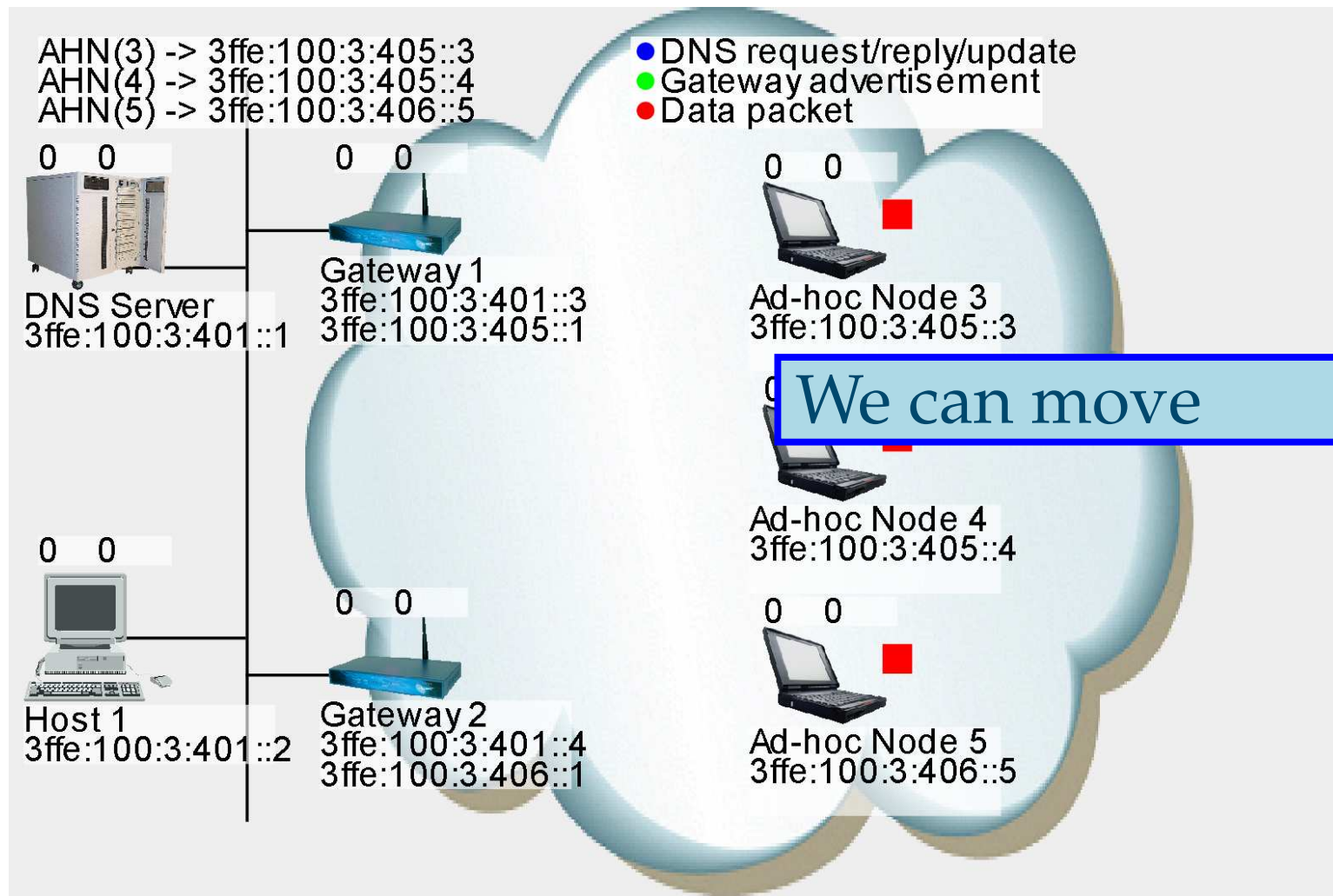
# The Project

- Participants: Ericsson Telebit, CPN Group at AU
- Project duration: July 2003 - December 2005
- Executive summary summary: *This projects deals with the design and validation of routing protocols and other protocols in ad-hoc and mobile networks*
- Project home-page:  
<http://www.daimi.au.dk/CPnets/IPv6/>

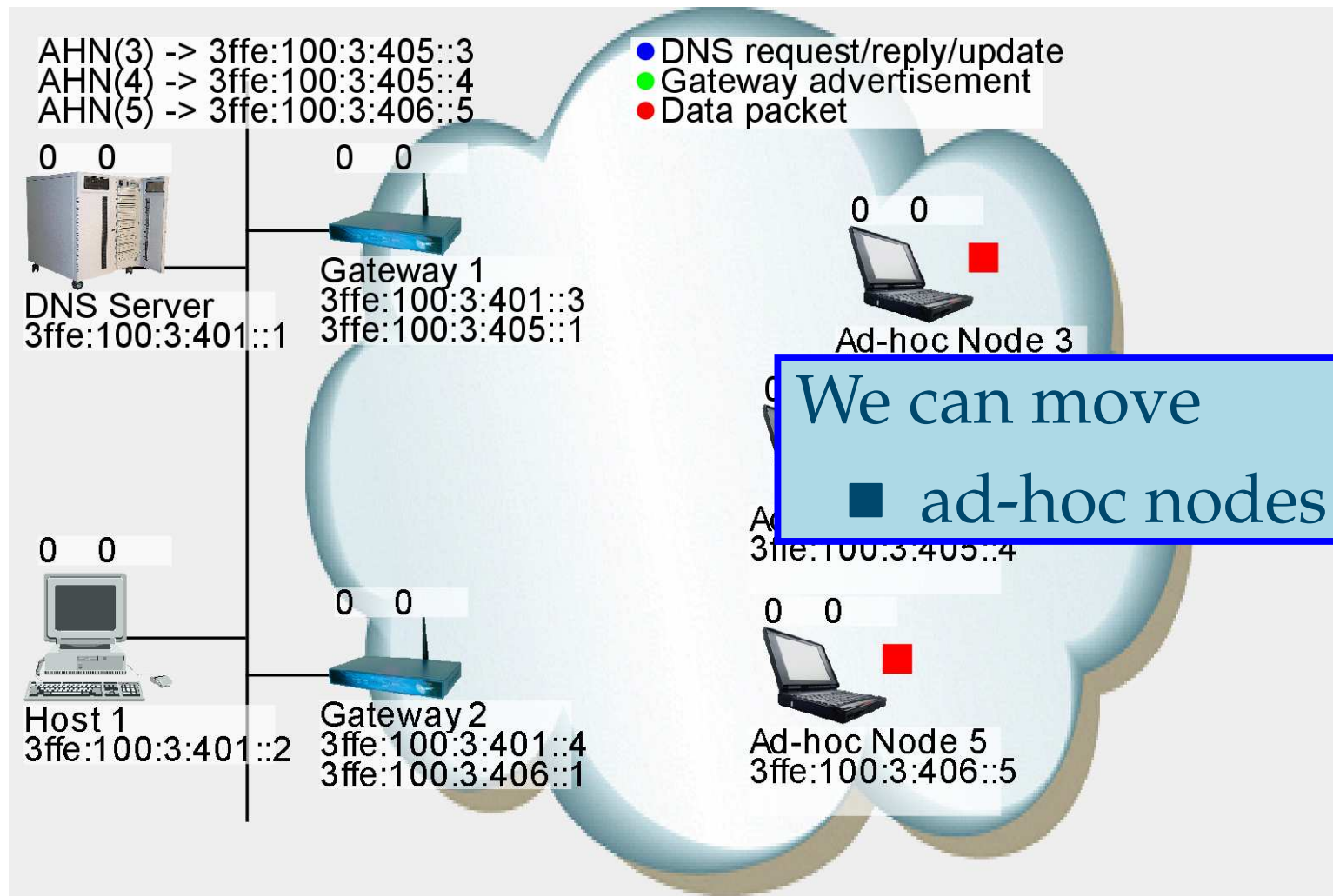
# Mobile Ad-hoc Networks



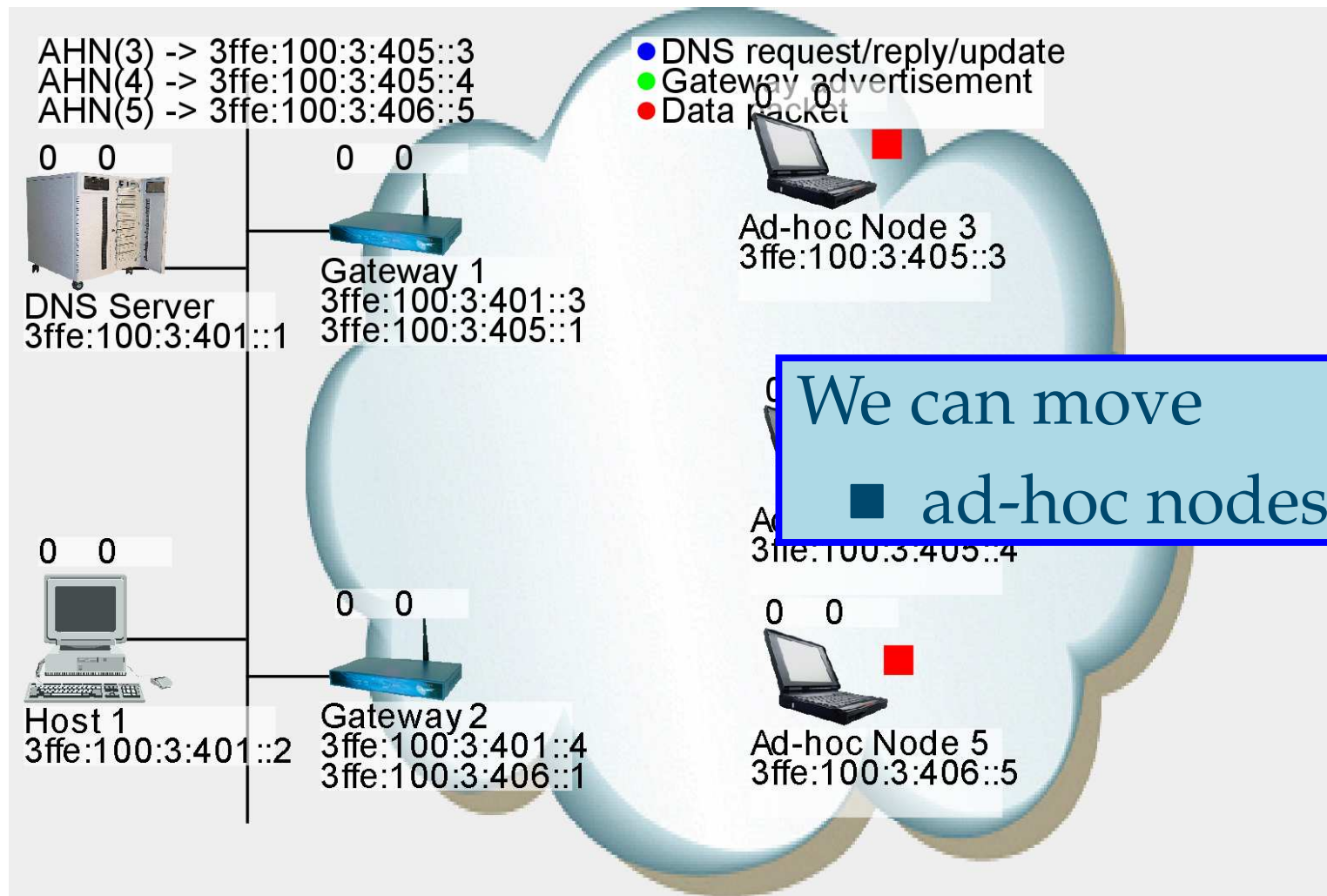
# Mobile Ad-hoc Networks



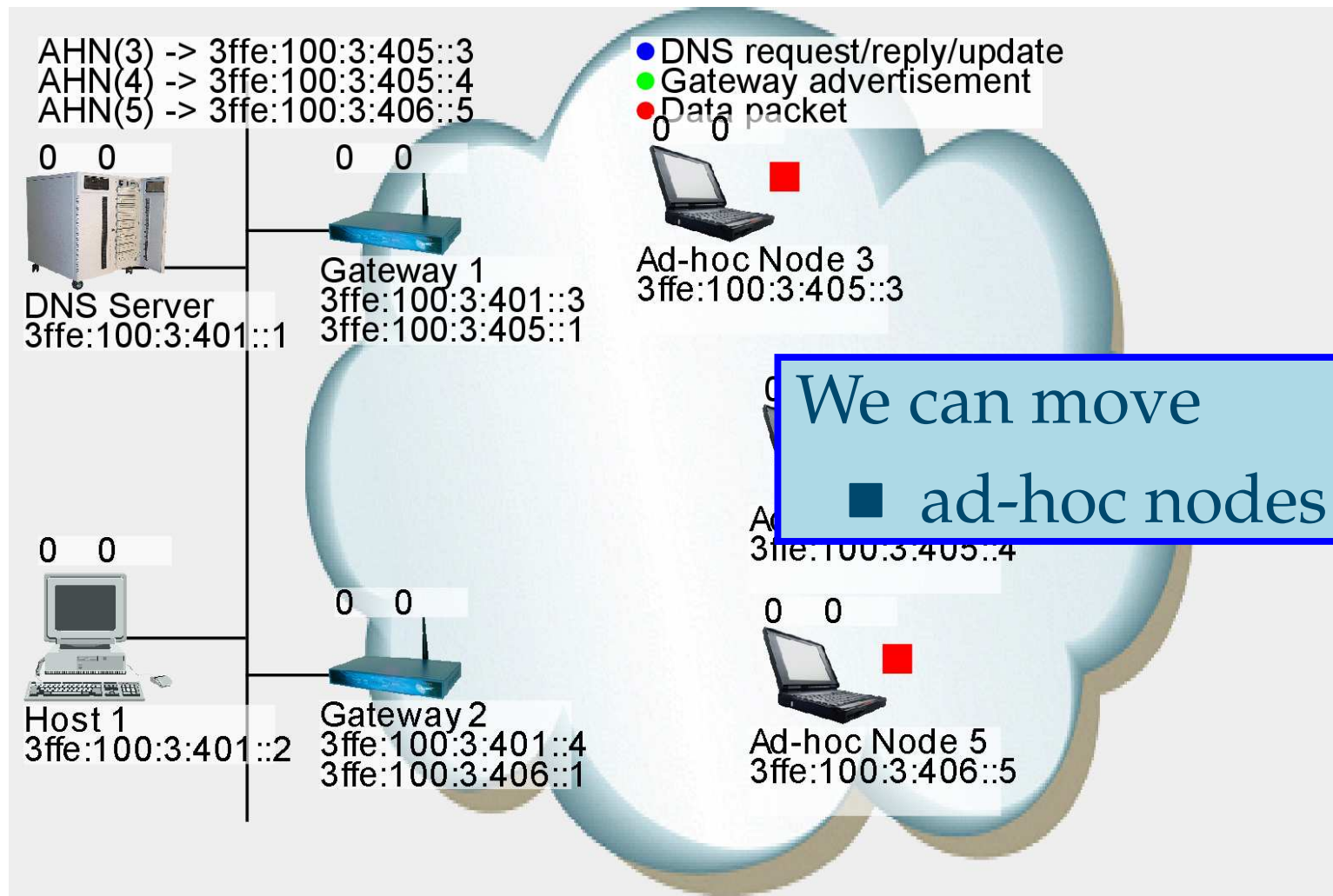
# Mobile Ad-hoc Networks



# Mobile Ad-hoc Networks

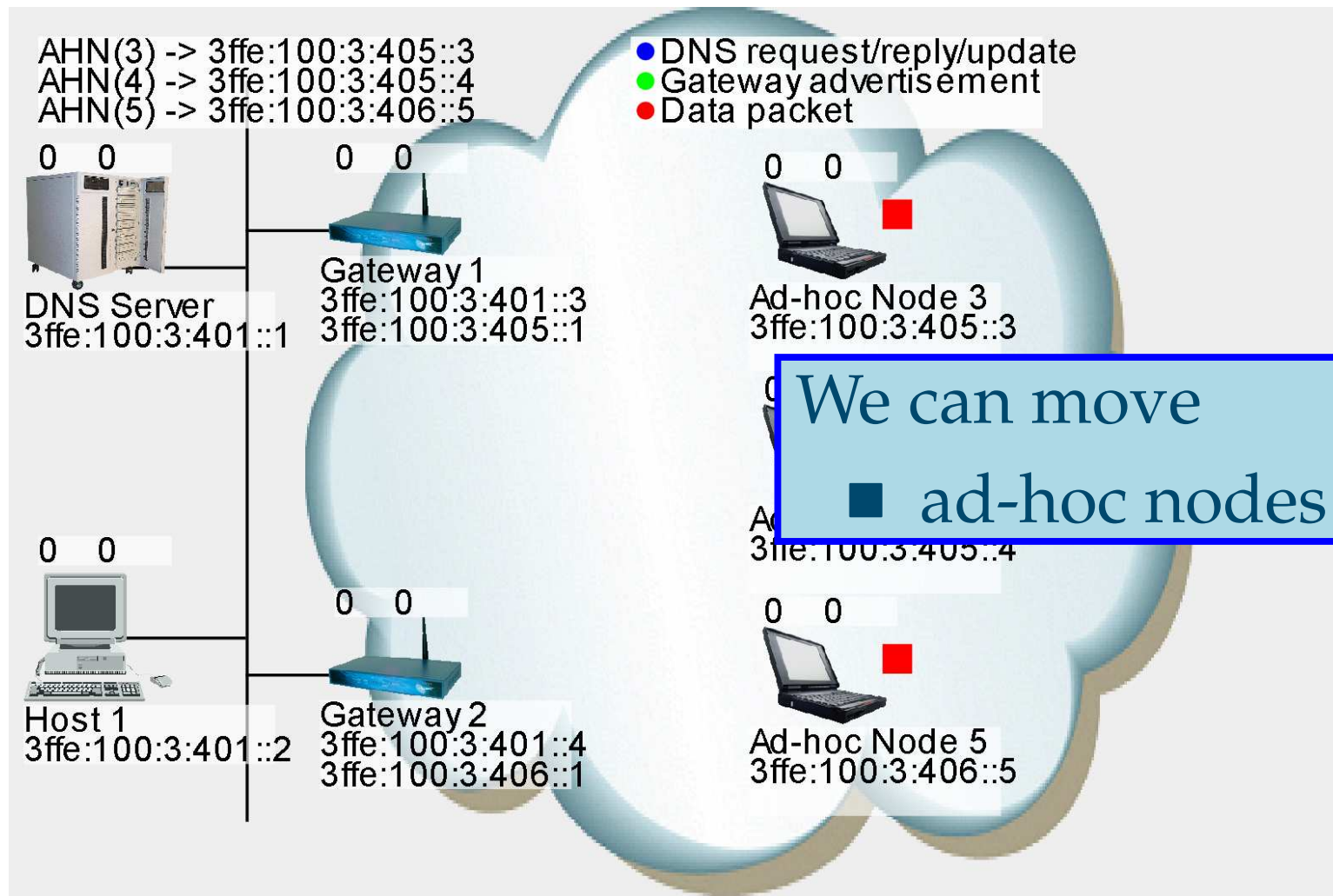


# Mobile Ad-hoc Networks

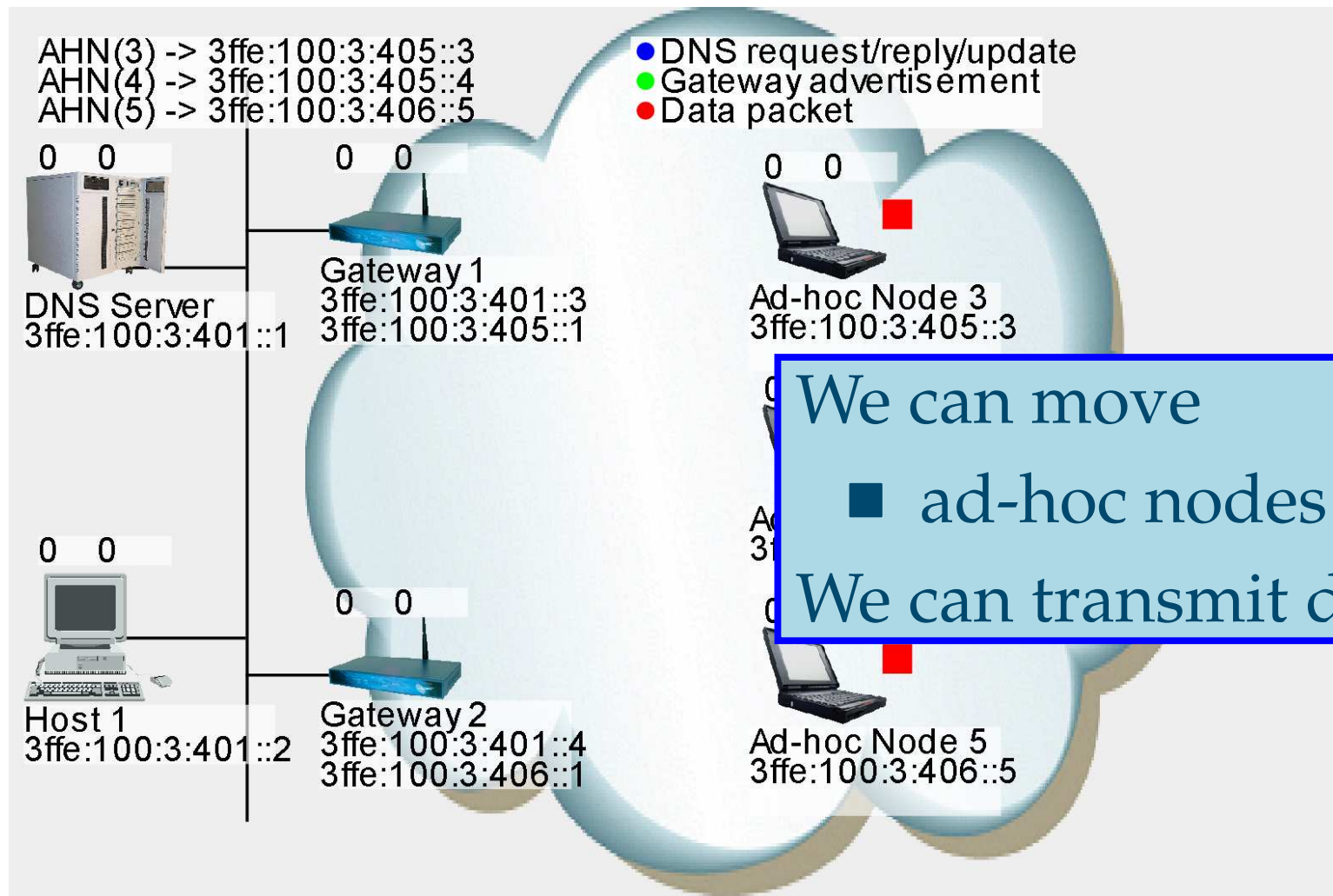




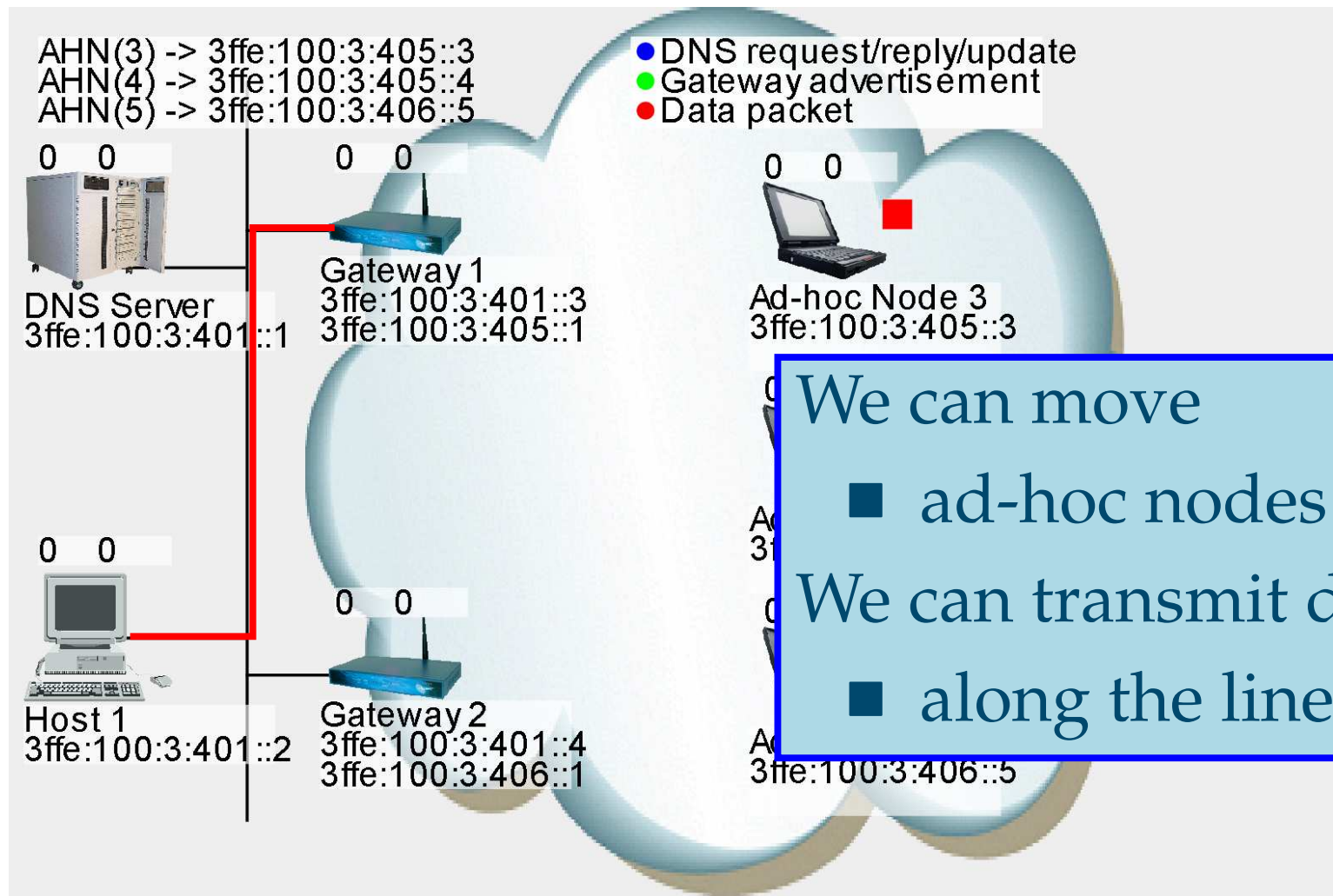
# Mobile Ad-hoc Networks



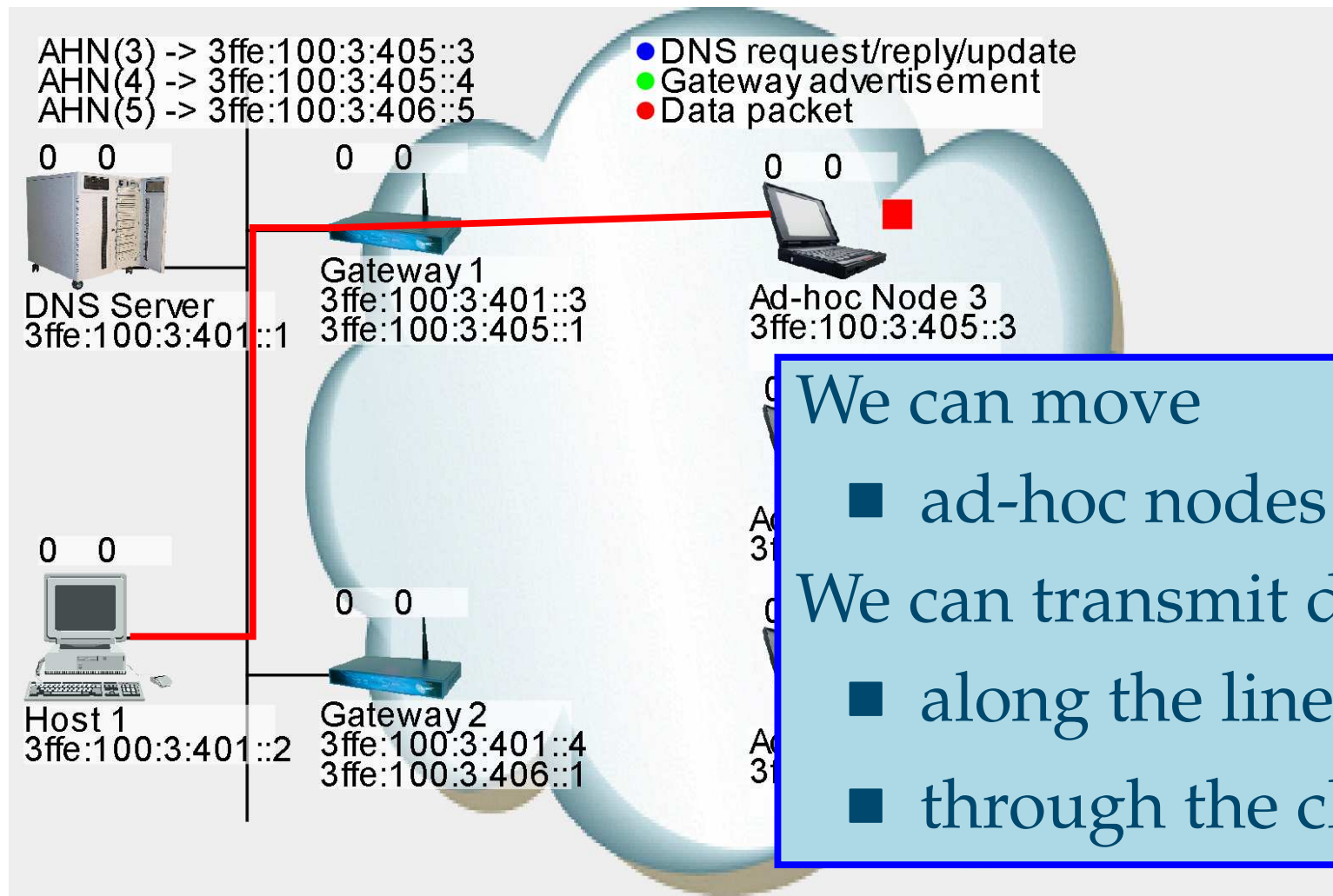
# Mobile Ad-hoc Networks



# Mobile Ad-hoc Networks



# Mobile Ad-hoc Networks



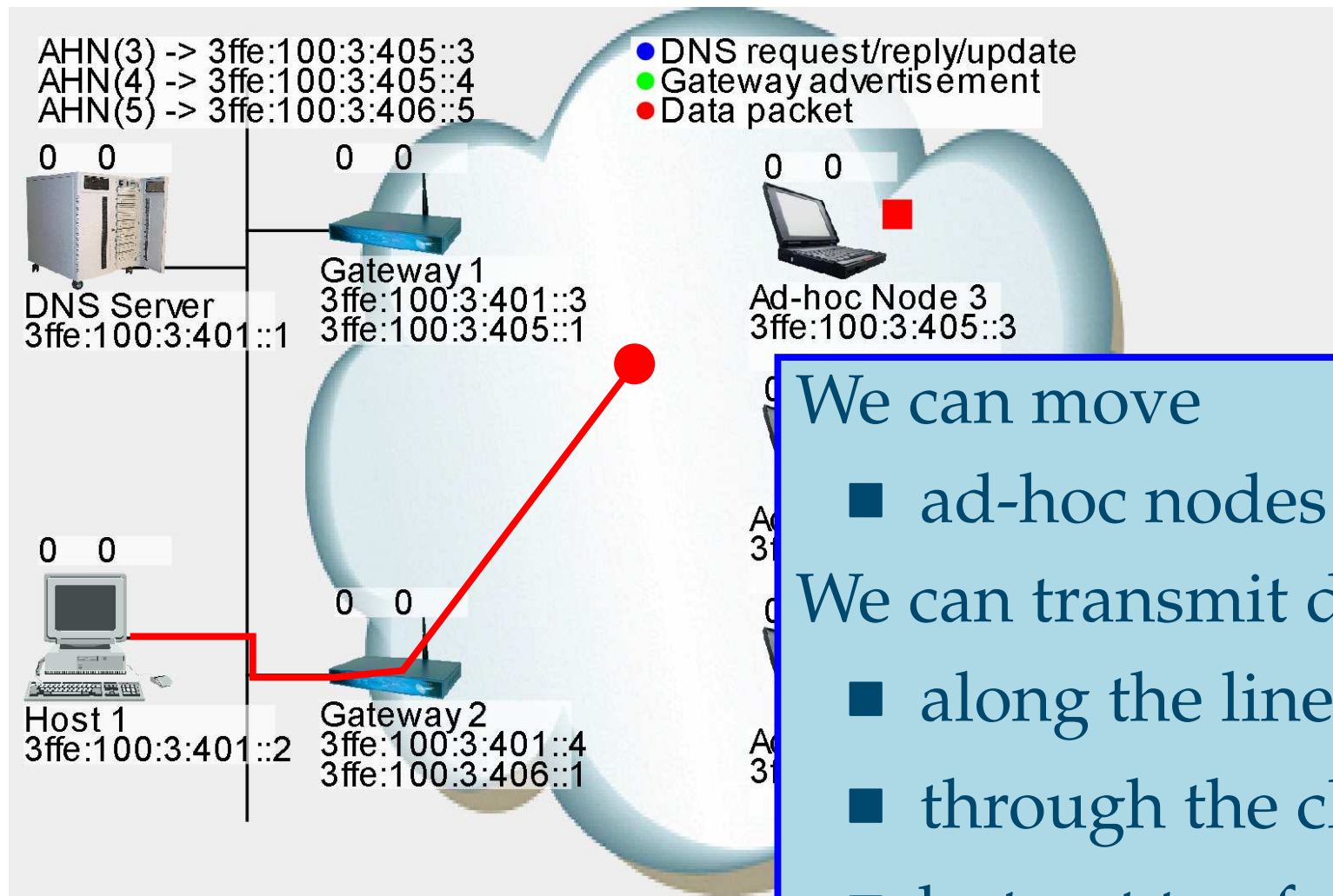
We can move

- ad-hoc nodes

We can transmit data

- along the lines
- through the cloud

# Mobile Ad-hoc Networks



We can move

- ad-hoc nodes

We can transmit data

- along the lines
- through the cloud
- but not too far

# Real-life Application

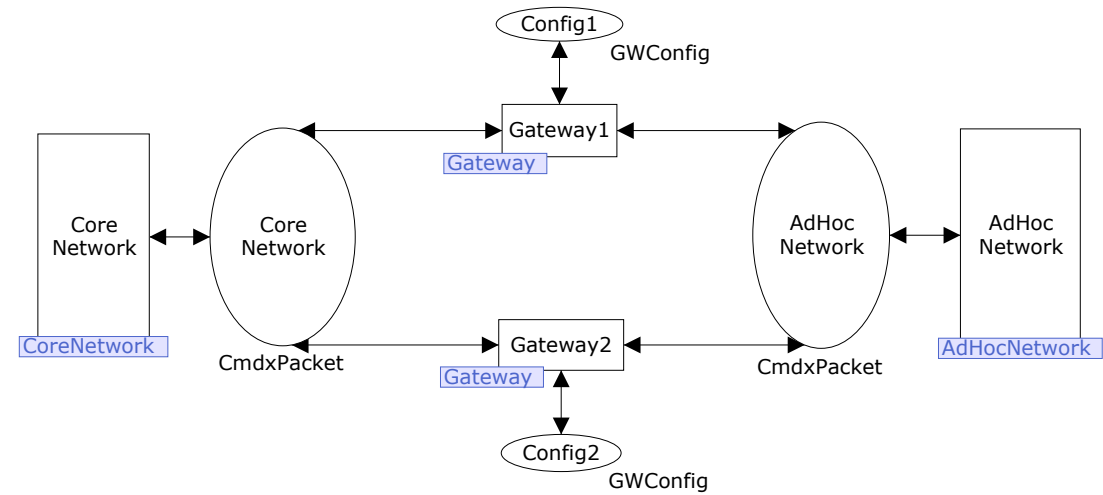
- Modern mobile phone (GPRS or 3G) connected to a service provider, e.g. streaming music
- Sitting in a moving train
- Want the music to play continuously – even when moving from one antenna to another



# CPN Model

```
gassign_ip("gw1", "3ffe:100:3:401::3",  
"3ffe:100:3:401::3", "3ffe:100:3:401::3")
```

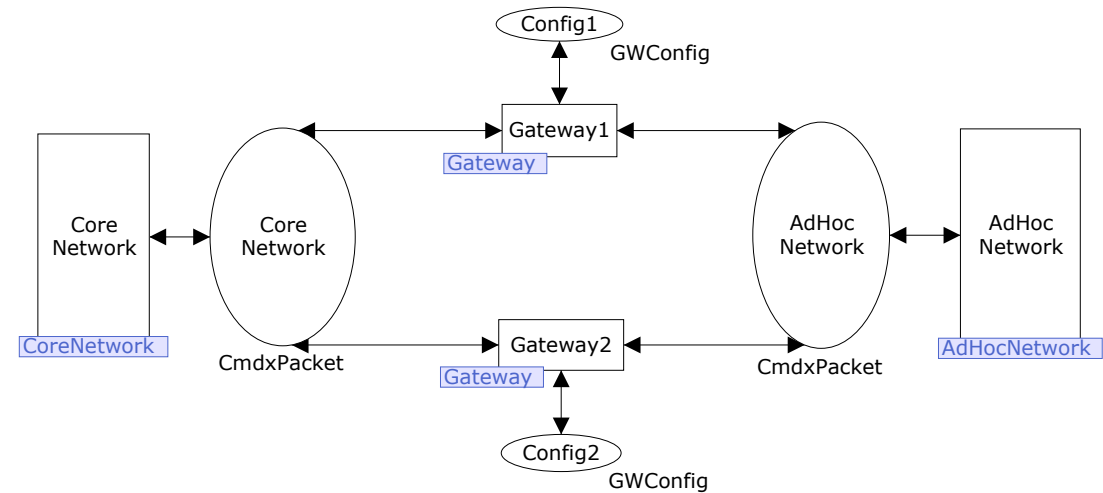
- 54 places
- 40 transitions



# CPN Model

```
gassign_ip("gw1", "3ffe:100:3:401::3",  
"3ffe:100:3:401::3", "3ffe:100:3:401::3")
```

- 54 places
- 40 transitions
- = big

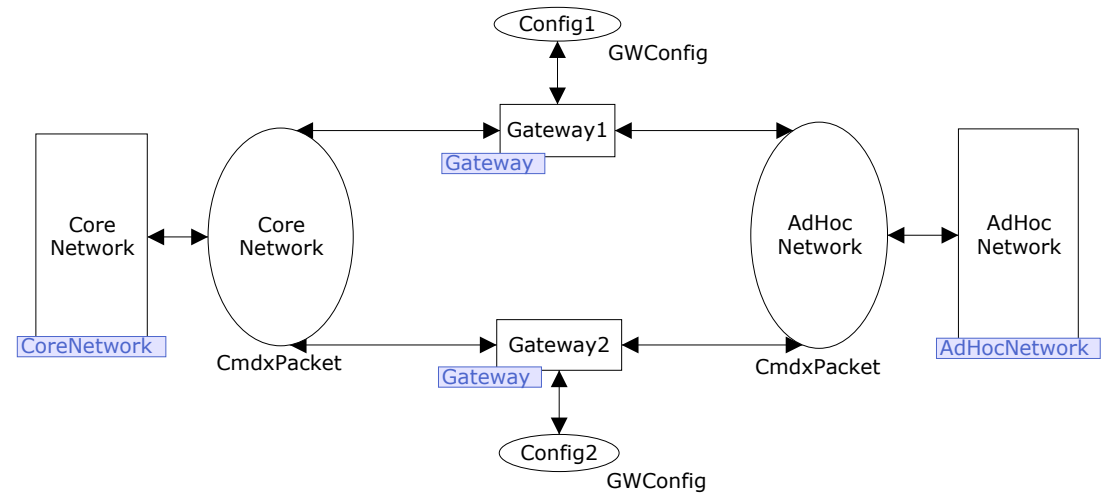




# CPN Model

```
gwassign_ip("gw1", "3ffe:100:3:401::3",  
"3ffe:100:3:401::3", "3ffe:100:3:401::3")
```

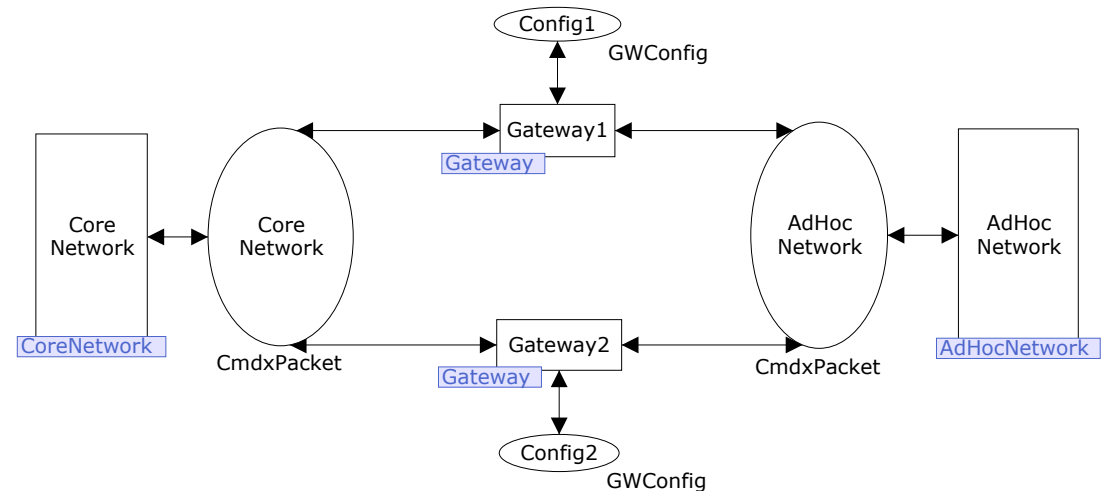
- 54 places
- 40 transitions
- = big
- = incomprehensible?



# CPN Model

```
gwassign_ip("gw1", "3ffe:100:3:401::3",
```

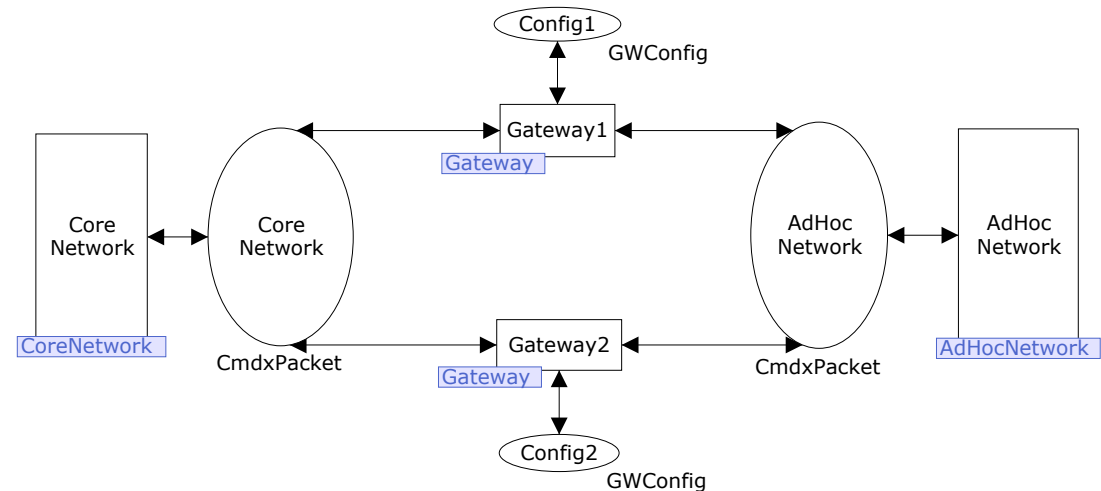
- 54 places
- 40 transitions
- = big
- = incomprehensible?
- No because we use modules



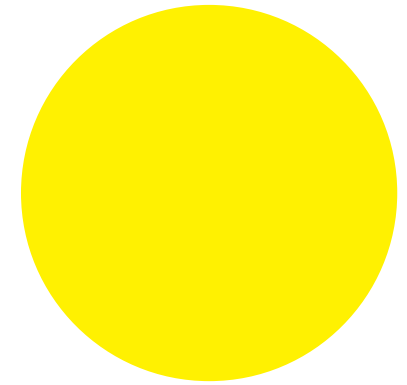
# CPN Model

```
gwassign_ip("gw1", "3ffe:100:3:401::3",  
"3ffe:100:3:401::3", "3ffe:100:3:401::3")
```

- 54 places
- 40 transitions
- = big
- = incomprehensible?
- No because we use modules
- 19 different modules

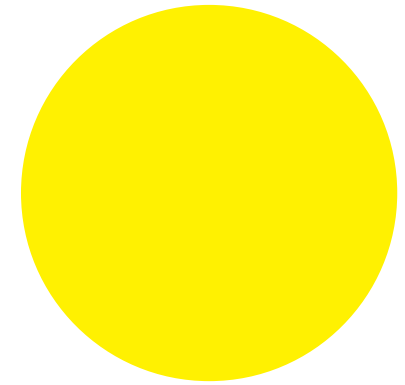


# A Java Program



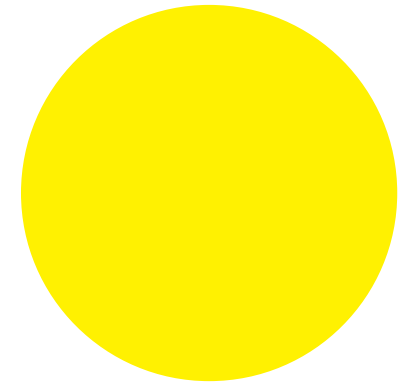
```
1 int radius = 5
2 int exponent = 2;
3 int radiusSquare = 1;
4 for (int i = 0; i < exponent; i++) {
5     radiusSquare = radiusSquare * radius
6 }
7 double area = radiusSquare * 3.1415926
```

# A Java Program



```
1 int radius = 5
2 int exponent = 2;
3 int radiusSquare = 1;
4 for (int i = 0; i < exponent; i++) {
5     radiusSquare = radiusSquare * radius
6 }
7 double area = radiusSquare * 3.1415926
```

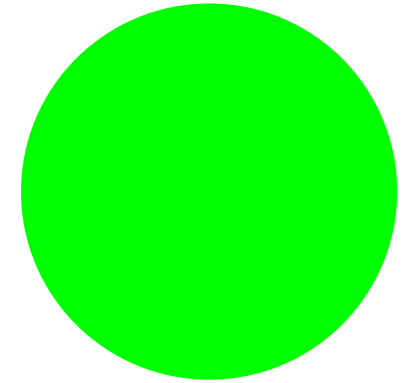
# A Java Program



```
1 int radius = 5
2 int exponent = 2;
3 int radiusSquare = 1;
4 for (int i = 0; i < exponent; i++) {
5     radiusSquare = radiusSquare * radius
6 }
7 double area = radiusSquare * 3.1415926
```

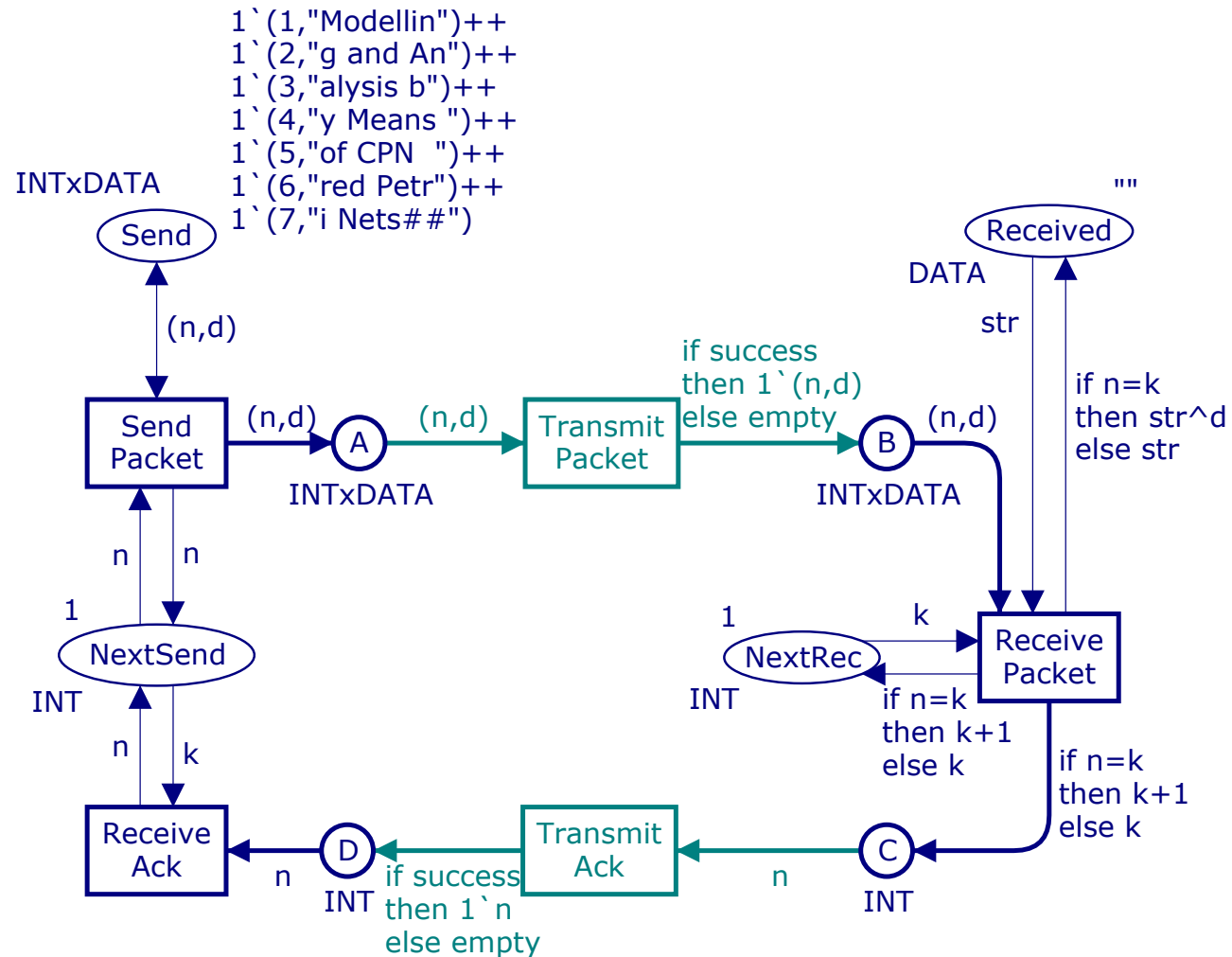
Not really part of  
the main flow

# A Simpler Java Program



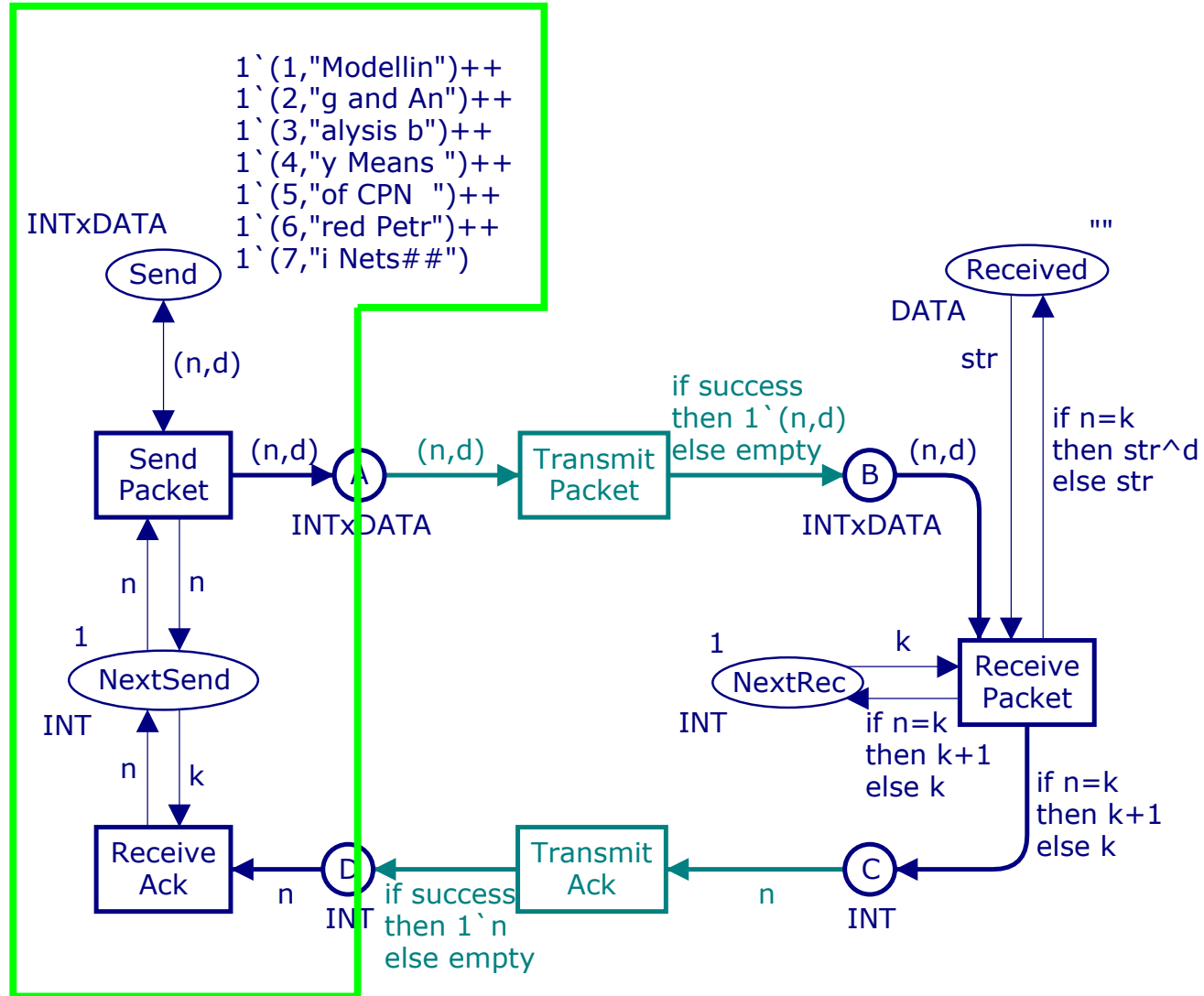
```
1  int power(int value , int exponent) {  
2      int result = 1;  
3      for (int i = 0; i < exponent; i++) {  
4          result = result * value;  
5      }  
6      return result;  
7  }  
8  
9  int radius = 5  
10 double area = power(radius , 2) * 3.141592
```

# A Coloured Petri-net

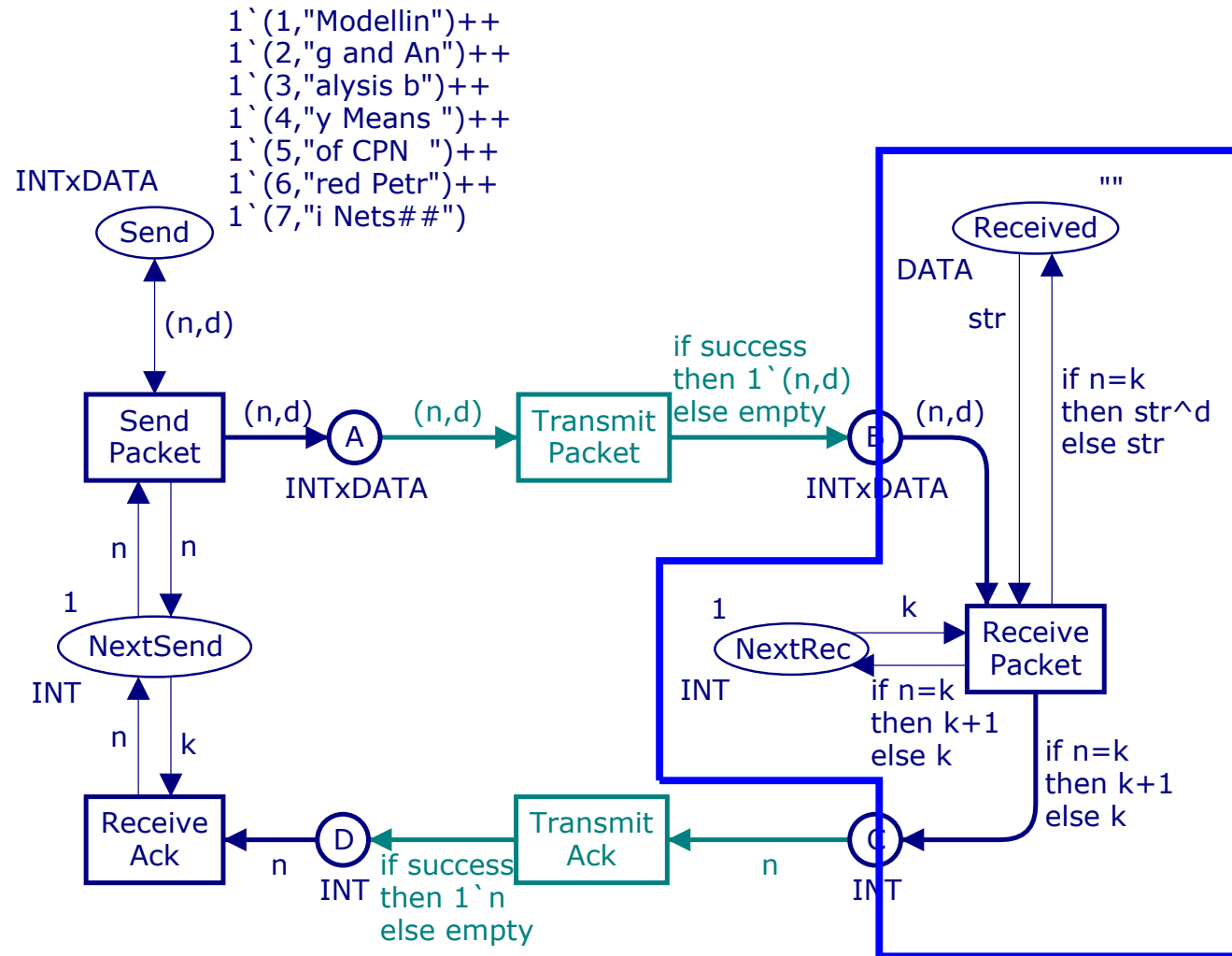




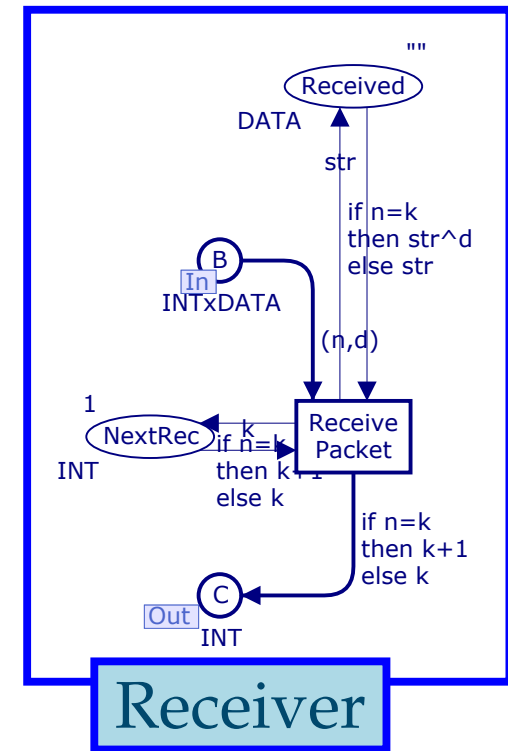
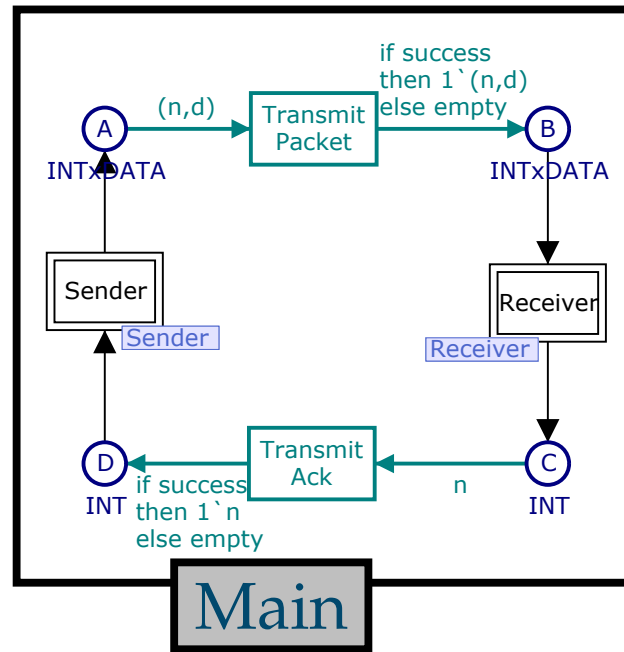
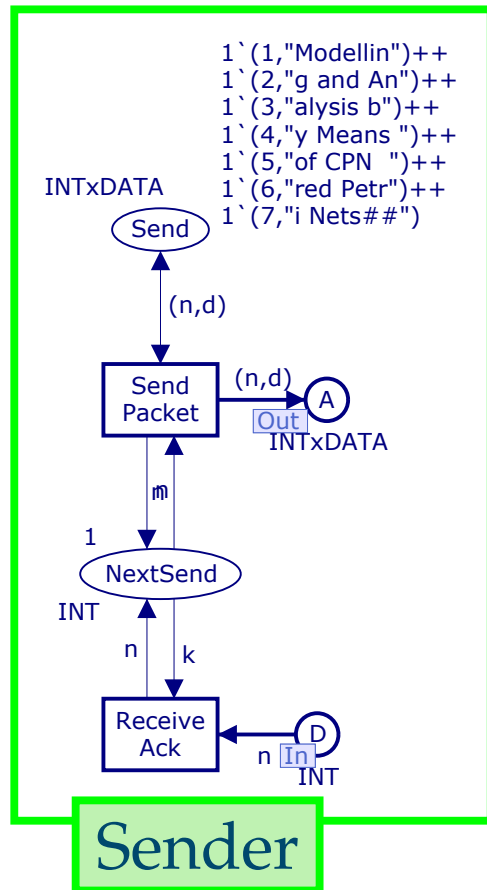
# A Coloured Petri-net



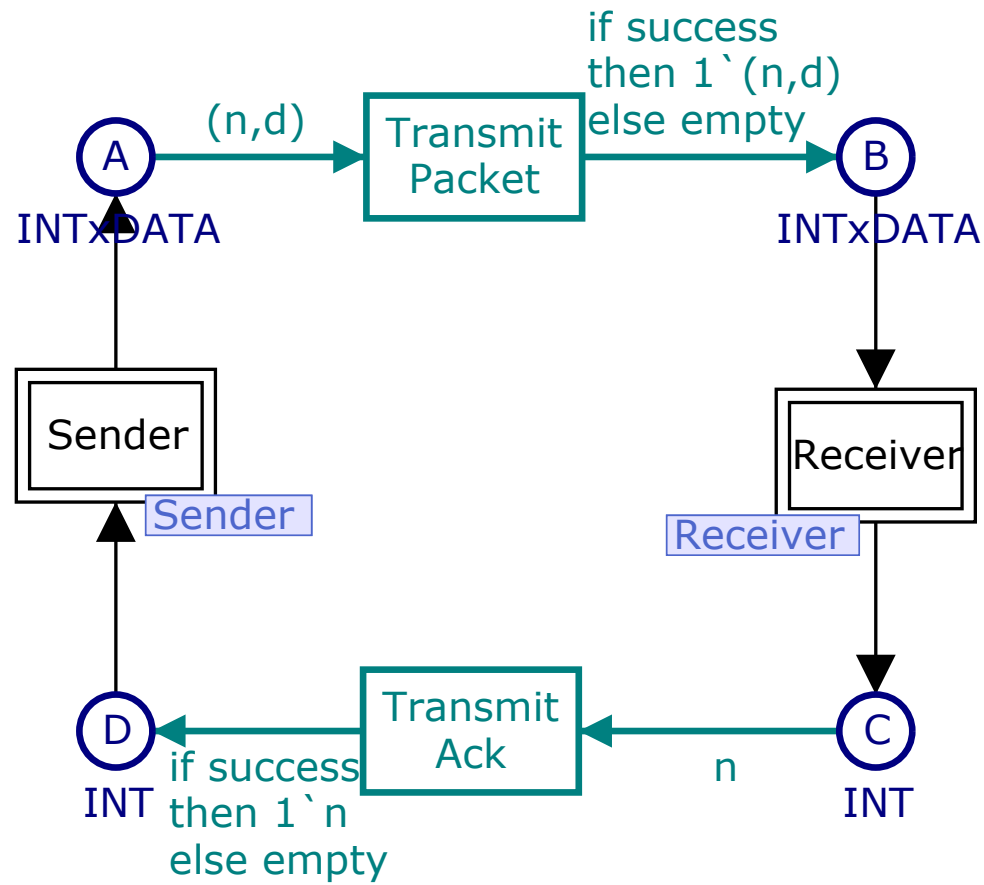
# A Coloured Petri-net



# A Simpler Coloured Petri-net



# Main Module

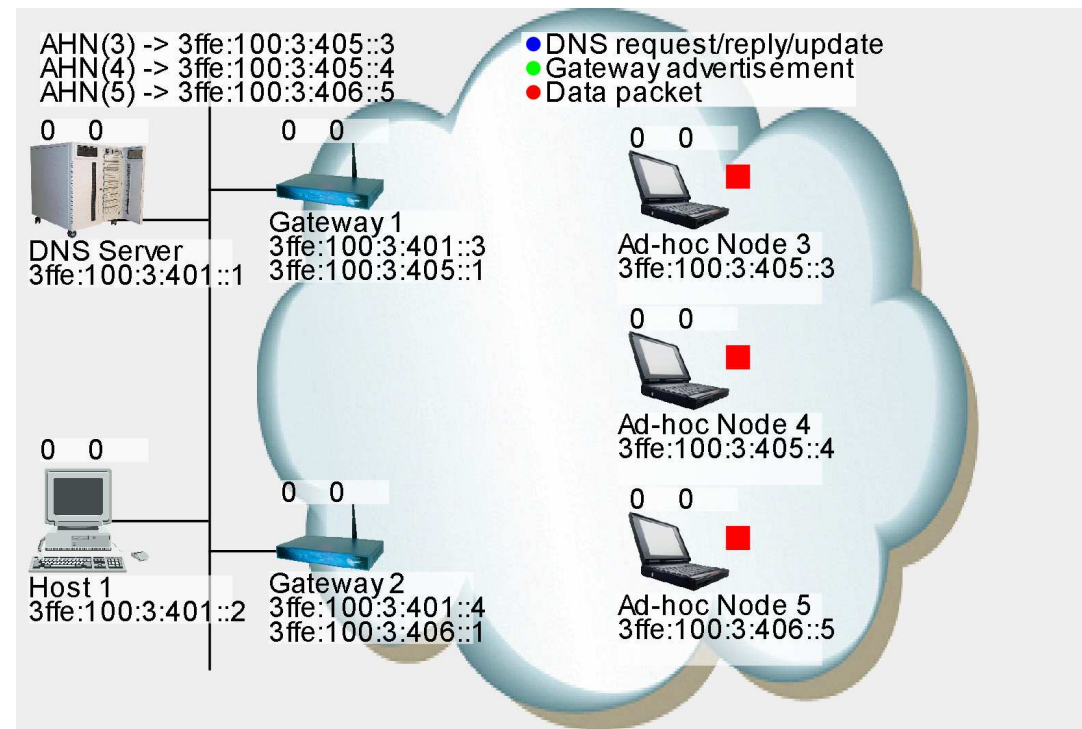


# Advantages of Modules

- We can split up our program/model into smaller, more comprehensible parts
- We can re-use sub-modules  
`c = squareRoot(power(a, 2) + power(b, 2));`
- We can replace submodules

```
1 int power(int v, int e) {
2     return (int) Math.pow(v, e);
3 }
```

# Small Scenario



- Host 1 wants to send data to Ad-hoc Node 3

# Demo

# Conclusions

- Coloured Petri-nets can cope with large, realistic models if we use modules
- The industry is interested in models
  - ◆ Easier to control and reproduce scenarios
  - ◆ Implementation details can be abstracted away