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

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

Ad-hoc Node 5
3ffe:100:3:406::5

Host 1
3ffe:100:3:401::2

Gateway 2
3ffe:100:3:401::4
3ffe:100:3:406::1
Mobile Ad-hoc Networks

We can move
Mobile Ad-hoc Networks

We can move
- ad-hoc nodes

AHN(3) -> 3ffe:100:3:405::3
AHN(4) -> 3ffe:100:3:405::4
AHN(5) -> 3ffe:100:3:406::5

DNS Server
3ffe:100:3:401::1

Host 1
3ffe:100:3:401::2

Gateway 1
3ffe:100:3:401::3
3ffe:100:3:405::1

Gateway 2
3ffe:100:3:401::4
3ffe:100:3:406::1

Ad-hoc Node 3
3ffe:100:3:405::4

Ad-hoc Node 5
3ffe:100:3:406::5
Mobile Ad-hoc Networks

We can move
- ad-hoc nodes
Mobile Ad-hoc Networks

We can move
- ad-hoc nodes
We can move

- ad-hoc nodes
Mobile Ad-hoc Networks

- We can move ad-hoc nodes
- We can transmit data
Mobile Ad-hoc Networks

- We can move ad-hoc nodes
- We can transmit data along the lines
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

- DNS request/reply/update
- Gateway advertisement
- Data packet
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

- 54 places
- 40 transitions
CPN Model

- 54 places
- 40 transitions
- = big
- 54 places
- 40 transitions
- = big
- = incomprehensible?
- 54 places
- 40 transitions
- = big
- = incomprehensible?
- No because we use modules
- 54 places
- 40 transitions
- = big
- = incomprehensible?
- No because we use modules
- 19 different modules
1 \textbf{int} \ radius = 5 \\
2 \textbf{int} \ radiusSquare = 1; \\
3 \textbf{for} \ (\textbf{int} \ i = 0; i < \text{radius}; i++) \ { \\
4 \hspace{1em} \text{radiusSquare} = \text{radiusSquare} \ast \text{radius} \\
5 \} \\
6 \textbf{double} \ \text{area} = \text{radiusSquare} \ast 3.1415926
```java
int radius = 5
int radiusSquare = 1;
for (int i = 0; i < radius; i++) {
    radiusSquare = radiusSquare * radius
}
double area = radiusSquare * 3.1415926
```
```java
int radius = 5
int radiusSquare = 1;
for (int i = 0; i < radius; i++) {
    radiusSquare = radiusSquare * radius
}
double area = radiusSquare * 3.1415926
```
A Simpler Java Program

```java
int square(int value) {
    int result = 1;
    for (int i = 0; i < value; i++) {
        result = result * value;
    }
    return result;
}

int radius = 5
double area = square(radius) * 3.1415926
```
A Coloured Petri-net
A Coloured Petri-net
A Coloured Petri-net
A Simpler Coloured Petri-net

Main

while_true

Process.all()

Enter Critical

b

ProcessXBool

1'(p(1),false) ++
1'(p(2),false)

critical

Process

Leaves Critical

Leave Critical

Enter

Enter Critical

(p(1),b)

ProcessXBool

bitrue

Process

processall()

k

Process

([34] p(0))

whilebjk

if k then
1' x(i)
else empty
[=3i]

whilebj

if k then
1' x(i)
else empty
[=3i]

whilek

if k then
1' x(i)
else empty
[=3i]

whilek

[=3i]
Advantages of Modules

- We can split up our program/model into smaller, more comprehensible parts
- We can re-use sub-modules
  \[ c = \text{squareRoot}(\text{square}(a) + \text{square}(b)) \];
- We can replace submodules
  ```
  int square(int value) {
    return (int) Math.pow(value, 2);
  }
  ```
**Small Scenario**

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