SIMULACIÓN DE PROTOCOLOS DE ENRUTAMIENTO PARA REDES MÓVILES AD-HOC MEDIANTE HERRRAMIENTA DE SIMULACIÓN NS-3

Mobility in ns-3

Contenidos

1. Mobility models
   • Introduction
   • Random Walk Model
   • Random WayPoint Model
   • Gauss Markov Model
2. Model description in ns-3
   • Design
   • Mobility model Classes and subclasses
3. Examples
4. Tracing and Netanim
5. References
6. Exercises
Mobility models. Introduction

- A mobility model should attempt to mimic the movements of real mobile nodes.
  - Changes in speed and direction must occur in reasonable time slots.
  - They are important to evaluate protocols in ad hoc networks.

- Classification:
  - Entity mobility models in which mobile node movements are independent of each other.
  - Group mobility models in which mobile node movements are dependent on each other.
    - A group of soldiers in a military scenario
    - Police officers attempting to catch an escaped criminal

Random Walk Mobility Model
- A simple mobility model based on random directions and speeds

Random Waypoint Mobility Model
- A model that includes pause times between changes in destination and speed

Random Direction Mobility Model
- A model that forces MNs to travel to the edge of the simulation area before changing direction and speed.

A Boundless Simulation Area Mobility Model
- A model that converts a 2D rectangular simulation area into a torus-shaped simulation area.

Gauss-Markov Mobility Model
- A model that uses one tuning parameter to vary the degree of randomness in the mobility pattern.

City Section Mobility Model:
- A simulation area that represents streets within a city.
Mobility models. Introduction. Group mobility models

- Exponential Correlated Random Mobility Model
  - A group mobility model that uses a motion function to create movements.

- Column Mobility Model
  - A group mobility model where the set of MNs form a line and are uniformly moving forward in a particular direction.

- Nomadic Community Mobility Model
  - A group mobility model where a set of MNs move together from one location to another.

- Pursue Mobility Model
  - A group mobility model where a set of MNs follow a given target.

- Reference Point Group Mobility Model
  - A group mobility model where group movements are based upon the path traveled by a logical center.

Mobility models. Random Walk Mobility Model

- Einstein 1926
- Mimic erratic movement
- Memoryless because it retains no knowledge concerning its past location and speed values.
- A mobile node moves from its current location to a new location by randomly choosing a direction and speed in which to travel
- Each movement occurs in either a constant time interval or a constant distance
- If a mobile reaches a simulation boundary, it bounces off the simulation border with an angle determined by the incoming direction.
- Simplified by assigning the same speed to every mobile node
Mobility models. Random Walk Mobility Model

- **Traveling patterns** [Camp 2002]

If the time or distance is short, the traveling patterns are random patterns restricted to a small portion of the simulation area.

Mobility models. Random Waypoint Mobility Model

- Bettsetter 2003
- The most used
- Memoryless because it retains no knowledge concerning its past location and speed values
- It includes pause times between changes in direction and/or speed
  - A mobile node begins by staying in one location for a certain period of time (pause time).
  - A random destination in the simulation area and an uniformly distributed speed is choosen between $[\text{min\_speed}, \text{max\_speed}]$
  - Upon arrival, the mobile pauses before starting the process again
Mobility models. Random Waypoint Mobility Model

- Traveling pattern
  - Initial localization (133,180); speed Uniform[0,10m/s]
- Similar to Random Walk if pause is zero.

- High variability in average mobile neighbor percentage during the initialization
  - Recommendation: long simulation runs

Mobility models. Gauss-Markov

- Tolety 1999
- Memory model
- Initially each mobile is assigned a current speed and direction
- At fixed intervals of time, $n$, movement occurs by updating the speed and direction of each mobile
- Speed, $s_n$, and direction, $d_n$, at time $n$ depends on $n-1$ using the following equations:
  \[
  s_n = \alpha s_{n-1} + (1-\alpha)\bar{s} + \sqrt{(1-\alpha^2)}s_{n-1} \quad d_n = \alpha d_{n-1} + (1-\alpha)\bar{d} + \sqrt{(1-\alpha^2)}d_{n-1}
  \]
- $\alpha$ [0,1] is the tuning parameter used to vary the randomness
  - $\alpha = 0$ random values; $\alpha = 1$ lineal motions
- At each time interval the next location is calculated based on the current location, speed, and direction of movement.
  \[
  x_n = x_{n-1} + s_{n-1} \cos d_{n-1} \quad y_n = y_{n-1} + s_{n-1} \sin d_{n-1}
  \]
Mobility models. Gauss-Markov

- Gauss-Markov model can eliminate the sudden stops and sharp turns that appeared in the Random WayPoint Model.
- Drawback is more complex because it has more parameters.
- Traveling pattern

![Diagram](image]

Model description in ns-3
Model description in ns-3

- The mobility support in |ns3| includes:
  - A set of mobility models which are used to track and maintain the cartesian position and speed of an object.
  - A "course change notifier" trace source which can be used to register listeners to the course changes of a mobility model.
  - A number of helper classes which are used to place nodes and setup mobility models.

- The source code for mobility lives in the directory `src/mobility`

- Information:
  - [http://www.nsnam.org/docs/release/3.18/doxygen/group__mobility.html](http://www.nsnam.org/docs/release/3.18/doxygen/group__mobility.html)

- The design includes mobility models, position allocators, and helper functions.

---

Model description in ns-3. Design: MobilityModel

- `MobilityModel` objects track the evolution of position with respect to a cartesian coordinate system.
  - Nowadays ns3 uses the Cartesian coordinate system only.

- The mobility model is typically aggregated to an `ns3::Node` object.

- The base class `ns3::MobilityModel` is subclassed for different motion behaviors.
Model description in ns-3. Design: Position Allocators

- The initial position of objects is typically set with a PositionAllocator.
- These types of objects will lay out the position on a canvas.
- Position allocators usually used only at beginning, to lay out the nodes initial position. However, some mobility models (e.g. RandomWaypoint) will use a position allocator to pick new waypoints.

Model description in ns-3. Design: Helpers

- Most users interact with the mobility system using mobility helper classes.
- The MobilityHelper combines a mobility model and position allocator, and can be used with a node container to install mobility capability on a set of nodes.
Model description in ns-3. Classes and Subclasses

- MobilityModel base class
  - GetPosition
  - Position and Velocity attributes
  - GetDistanceFrom
  - CourseChangeNotification

- MobilityModel Subclasses
  - ConstantPosition
  - ConstantVelocity
    - Mobility model for which the current speed does not change once it has been set and until it is set again explicitly to a new value.
  - ConstantAcceleration
    - Mobility model for which the current acceleration does not change once it has been set and until it is set again explicitly to a new value.

- MobilityModel Subclasses
  - GaussMarkov
  - RandomDirection2D
    - The movement of objects is based on random directions: each object pauses for a specific delay, chooses a random direction and speed and then travels in the specific direction until it reaches one of the boundaries of the model. When it reaches the boundary, it pauses, selects a new direction and speed.

- RandomWalk2D
  - Each instance moves with a speed and direction chosen at random with the user-provided random variables until either a fixed distance has been walked or until a fixed amount of time. If we hit one of the boundaries (specified by a rectangle), of the model, we rebound on the boundary with a reflexive angle and speed. This model is often identified as a brownian motion model.
MobilityModel Subclasses

- RandomWaypoint
  - Each object starts by pausing at time zero for the duration governed by the random variable "Pause". After pausing, the object will pick a new waypoint (via the Position Allocator) and a new random speed via the random variable "Speed", and will begin moving towards the waypoint at a constant speed. When it reaches the destination, the process starts over (by pausing).

- SteadyStateRandomWaypoint
  - This model based on random waypoint mobility (RWM) model for case when speed, pause and position are uniformly distributed random variables. The difference is that the initial values of this parameters are not from uniform distribution but from stationary distribution of RWM model. The implementation of this model is 2D-specific and with nonzero nodes speeds.

PositionAllocator

- ListPositionAllocator
- GridPositionAllocator
- RandomRectanglePositionAllocator (2D)
  - Allocate random positions within a rectangle according to a pair of random variables
- RandomBoxPositionAllocator (3D)
- RandomDiscPositionAllocator
  - Allocate random positions within a disc according to a given distribution for the polar coordinates of each node with respect to the provided center of the disc.
- UniformDiscPositionAllocator
  - Allocate the positions uniformly (with constant density) randomly within a disc
Examples

Model description in ns-3. Example 1

- Two fixed nodes.
  - `ConstantPositionMobilityModel` allows nodes to remain in a fixed position during the simulation

```cpp
#include "ns3/mobility-module.h"
NodeContainer c;
c.Create (2);
MobilityModeMobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (5.0, 0.0, 0.0));
mobility.SetPositionAllocator (positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (c);
```
Model description in ns-3. Example 2

- 2-D grid topology.
  - ConstantPositionMobilityModel allows nodes to remain in a fixed position during the simulation

```cpp
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
  "MinX", DoubleValue (0.0),
  "MinY", DoubleValue (0.0),
  "DeltaX", DoubleValue (distance_x),
  "DeltaY", DoubleValue (distance_y),
  "GridWidth", UintegerValue (5),
  "LayoutType",StringValue("RowFirst"));

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (c);
```
Model description in ns-3. Example 3

- RandomDiscPositionAllocator
  (main-random-walk.cc )
  
  NodeContainer c;
  c.Create (100);

  MobilityHelper mobility;
  mobility.SetPositionAllocator
  ("ns3::RandomDiscPositionAllocator",
   "X", StringValue ("100.0"),
   "Y", StringValue ("100.0"),
   "Rho", StringValue
   ("ns3::UniformRandomVariable[Min=0|Max=30]");

Model description in ns-3. Example 3

class RandomDiscPositionAllocator : public PositionAllocator
227 {
228 public:
229 static TypeId GetTypeId (void);
230 RandomDiscPositionAllocator ();
231 virtual ~RandomDiscPosition Allocator ();
233 void SetTheta (Ptr<RandomVariableStream> theta);
234 void SetRho (Ptr<RandomVariableStream> rho);
235 void SetX (double x);
236 void SetY (double y);
238 virtual Vector GetNext (void) const;
239 virtual int64_t AssignStreams (int64_t stream);
240 private:
241 P tr<RandomVariableStream> m_theta;
242 Ptr<RandomVariableStream> m_rho;
243 double m_x;
244 double m_y;
245 };
Model description in ns-3. Example 4

- **RandomWalk2dMobilityModel**

  ```cpp
  mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
      "Mode", StringValue ("Time"),
      "Time", StringValue ("2s"),
      "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]")",
      "Bounds", StringValue ("0|200|0|200"));
  mobility.InstallAll ()

  mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
      "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
  mobility.Install (wifiStaNodes);
  ```

Model description in ns-3. Example 5

- **Gauss Markov Mobility model**

  ```cpp
  MobilityHelper mobility;
  mobility.SetMobilityModel ("ns3::GaussMarkovMobilityModel",
      "Bounds", BoxValue (Box (0, 150000, 0, 150000, 0, 10000)),
      "TimeStep", TimeValue (Seconds (0.5)),
      "Alpha", DoubleValue (0.85),
      "MeanVelocity", StringValue ("ns3::UniformRandomVariable[Min=800|Max=1200]"),
      "MeanDirection", StringValue ("ns3::UniformRandomVariable[Min=0|Max=6.283185307]"),
      "MeanPitch", StringValue ("ns3::UniformRandomVariable[Min=0.05|Max=0.05]"),
      "NormalVelocity", StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=0.0|Bound=0.0]"),
      "NormalDirection", StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=0.2|Bound=0.4]"),
      "NormalPitch", StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=0.02|Bound=0.04]"));
  mobility.SetPositionAllocator ("ns3::RandomBoxPositionAllocator",
      "X", StringValue ("ns3::UniformRandomVariable[Min=0|Max=150000]"),
      "Y", StringValue ("ns3::UniformRandomVariable[Min=0|Max=150000]"),
      "Z", StringValue ("ns3::UniformRandomVariable[Min=0|Max=10000]"));
  mobility.Install (wifiStaNodes);
  ```
Tracing and Netanim

Model description in ns-3. Trace events

- Mobility model predefined course change trace source to originate trace events

```cpp
void CourseChange(std::string context, Ptr<const MobilityModel> model) {
  Vector position = model->GetPosition ();
  NS_LOG_UNCOND (context << " x = " << position.x << ",
                  y = " << position.y); }
```

- This code just pulls the position information from the mobility model and unconditionally logs the x and y position of the node
Model description in ns-3. Trace events

- CourseChange function has to be called every time the wireless node changes its position.
- It is done using the Config::Connect function.

```
std::ostringstream oss;
oss <<
  "\NodeList/" << wifiStaNodes.Get(nWifi - 1)->GetId () <<
  "/$ns3::MobilityModel/CourseChange";

Config::Connect (oss.str (), MakeCallback (&CourseChange));
```

- What we do here is to create a string containing the tracing namespace path of the event to which we want to connect.

```
/NodeList/7/$ns3::MobilityModel/CourseChange
```

Model description in ns-3. Example Trace events

- Every course change event on node seven will be hooked into our trace sink, which will in turn print out the new position.

```
/NodeList/7/$ns3::MobilityModel/CourseChange x = 10, y = 0
/NodeList/7/$ns3::MobilityModel/CourseChange x = 9.41539, y = -0.811313
/NodeList/7/$ns3::MobilityModel/CourseChange x = 8.46199, y = -1.11303
/NodeList/7/$ns3::MobilityModel/CourseChange x = 7.52738, y = -1.46869
/NodeList/7/$ns3::MobilityModel/CourseChange x = 6.67099, y = -1.98503
/NodeList/7/$ns3::MobilityModel/CourseChange x = 5.6835, y = -2.14268
/NodeList/7/$ns3::MobilityModel/CourseChange x = 4.70932, y = -1.91689
```
Model description in ns-3. Netanim

- NetAnim is an offline animator based on the Qt 4 toolkit.
- It currently animates the simulation using an XML trace file collected during simulation.
- It comes with ns-3 software (*netanim-3.103*).

- What does it do?
  - Animate packets over wired-links and wireless-links
  - Packet timeline with regex filter on packet meta-data.
  - Node position statistics with node trajectory plotting.
  - Print brief packet-meta data on packets

Model description in ns-3. Netanim. Installation

- Prerequisites
  - mercurial
  - QT4 development packages (recommended version 4.7)
- Debian/Ubuntu Linux distribution:
  ```
  apt-get install mercurial
  apt-get install qt4-dev-tools
  ```
- Building and Starting
  ```
  cd netanim
  make clean
  qmake NetAnim.pro
  make
  ```
- Using NetAnim. It is a two-step process.
  - Step 1: Generate the animation XML trace file during simulation using "ns3::AnimationInterface" in the ns-3 code base
  - Step 2: Load the XML trace file generated in Step 1 with the offline animator (NetAnim).
Model description in ns-3. Netanim. XML Generation

- They must be applied just before the "Simulation::Run" statement.

```cpp
#include "ns3/netanim-module.h"
[....]
AnimationInterface::SetNodeDescription (c, "ADHOC_NODO"); // Optional
AnimationInterface::SetNodeColor (c, 0, 255, 0); // Optional (verde)
AnimationInterface anim ("wifi-adhoc-animation.xml"); // Mandatory
anim.EnablePacketMetadata (true); // Optional
```

- Run the ns-3 code base to generate xml file
- Open the xml file with Netanim
  ```shell
cd netanim
  ./NetAnim &
  ```
  (Open the file from the Menu)

References

- Ns-3 tutorial and manual
- Netanim: http://www.nsnam.org/wiki/NetAnim
Exercises

Model description in ns-3. Exercise 1

- Model a Wireless Mesh Network (WMN) with 10 nodes
- It is static
- Initial position is a line with a distance between nodes of 200m
- Visualize with Netanim
Model description in ns-3. Exercise 2

- Model a mobile ad hoc network with 50 nodes.
- At the beginning they are situated randomly in a rectangle 1000*500
- Random Waypoint Mobility Model
- Visualize with Netanim