LEMMA, Modèles de Mobilité et Simulation

Alexander Pelov
Thomas Noël

RGE
jeudi, le 3 juin 2010

Plan

- Introduction
- LEMMA
- Simulateur LEMMA
- Conclusion

Introduction

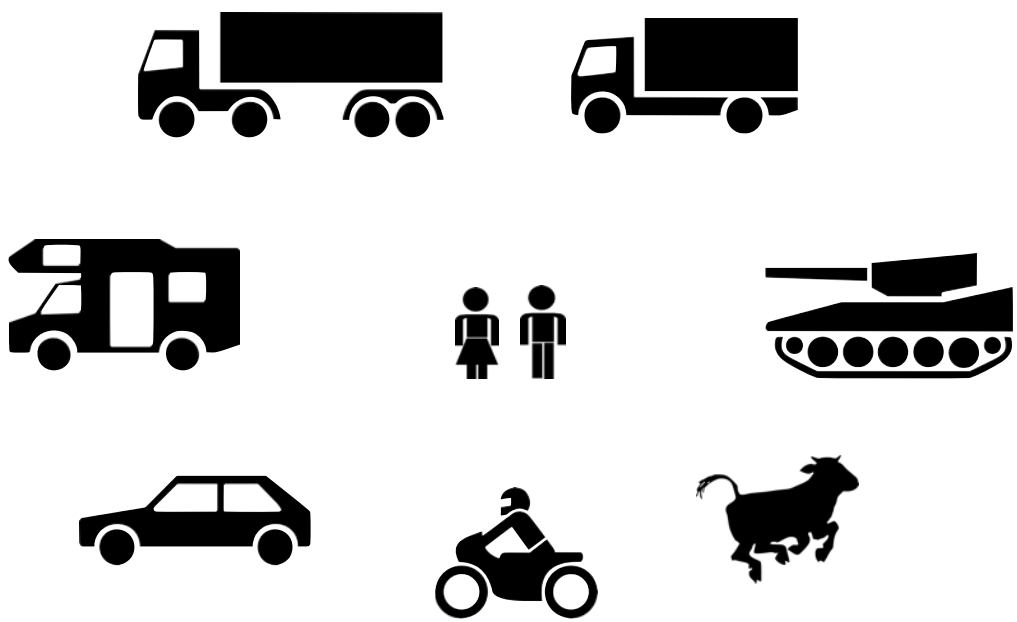
Réseaux sans fil

- Avec infrastructure, sans (ad hoc) ou mixte
- Connexions sans interruption vs. occasionnelles
- Réseaux de voitures, piétons, animaux, etc.
  - Différentes caractéristiques (vitesse de déplacement, énergie, ...)
- Equipements utilisés
  - Capteurs, téléphones et ordinateurs portables, i.e. terminaux de communication

Entités visées

Entités visées

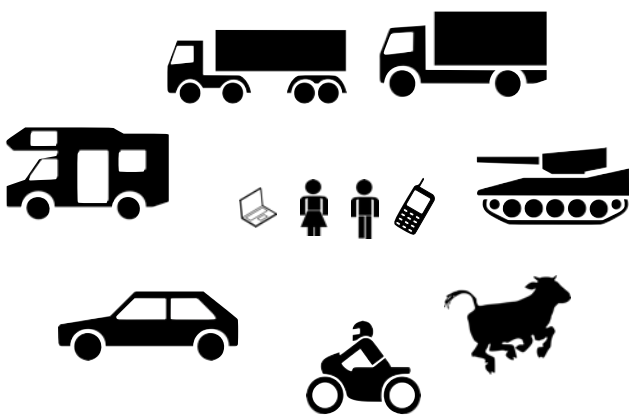
Entités visées



Entités visées

Terminologie

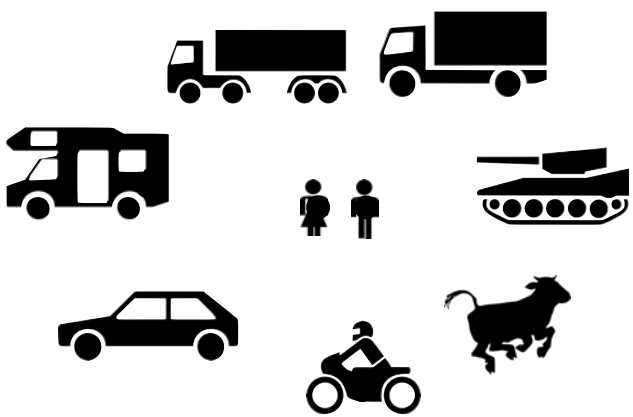
Entités



Nœuds

Terminologie

Entités

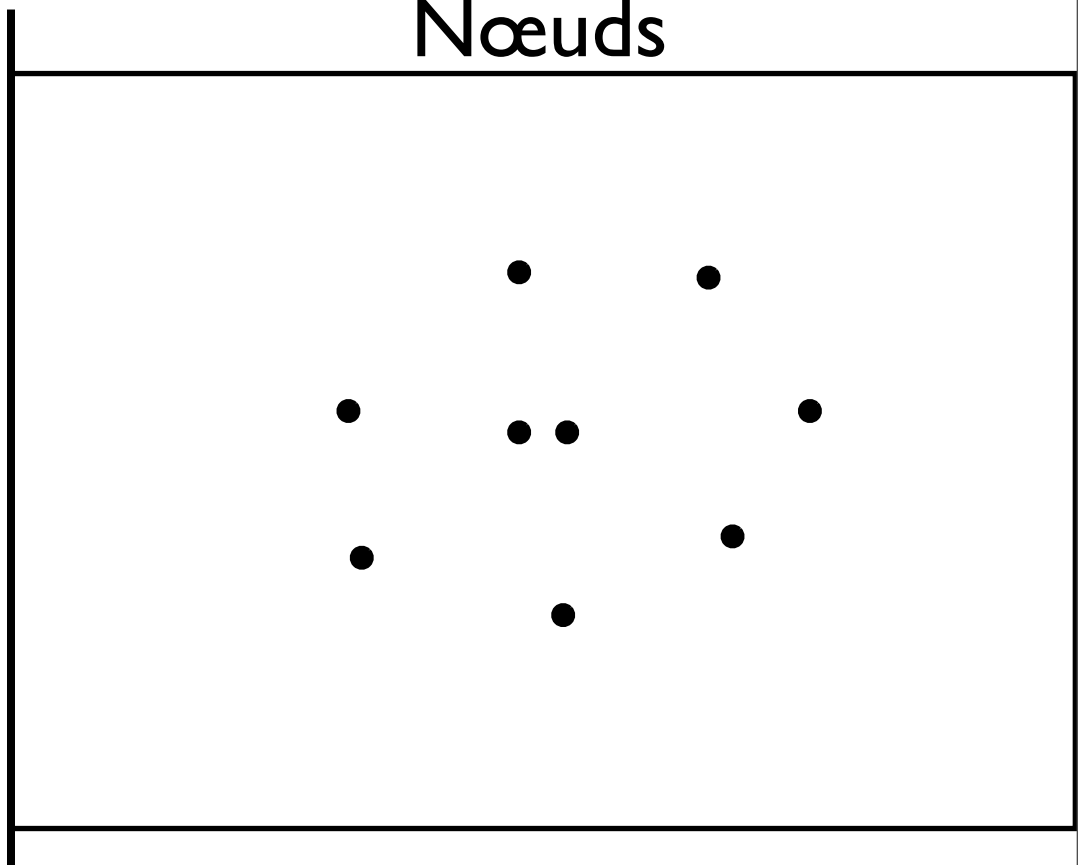


Nœuds

Terminologie

Entités

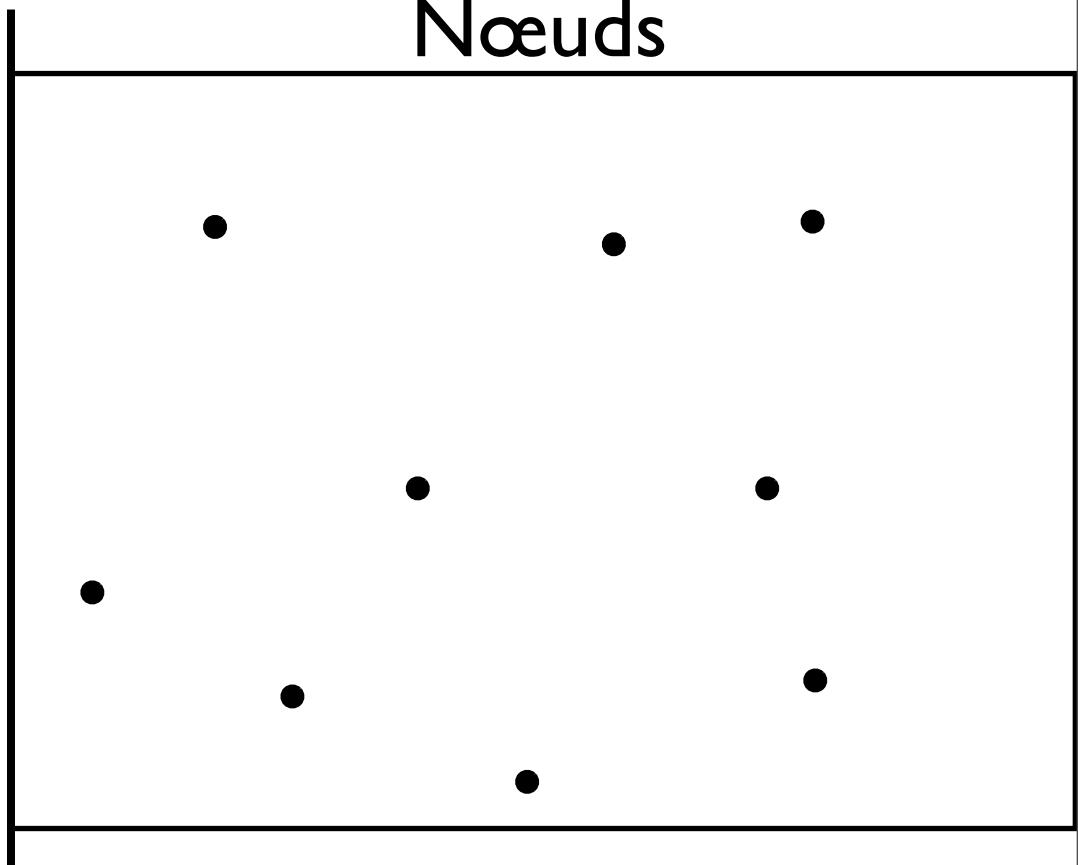
Nœuds



Terminologie

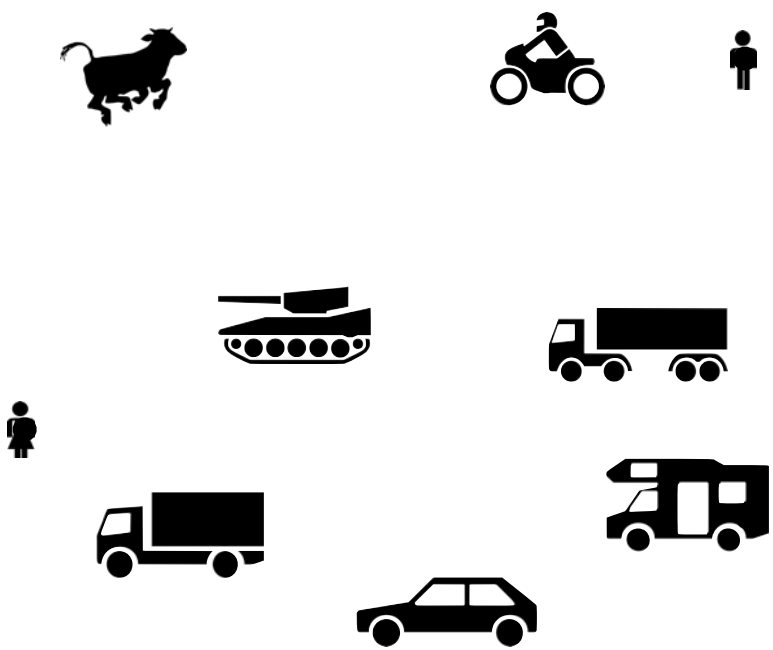
Entités

Nœuds



Terminologie

Entités



Nœuds

Modèle de mobilité

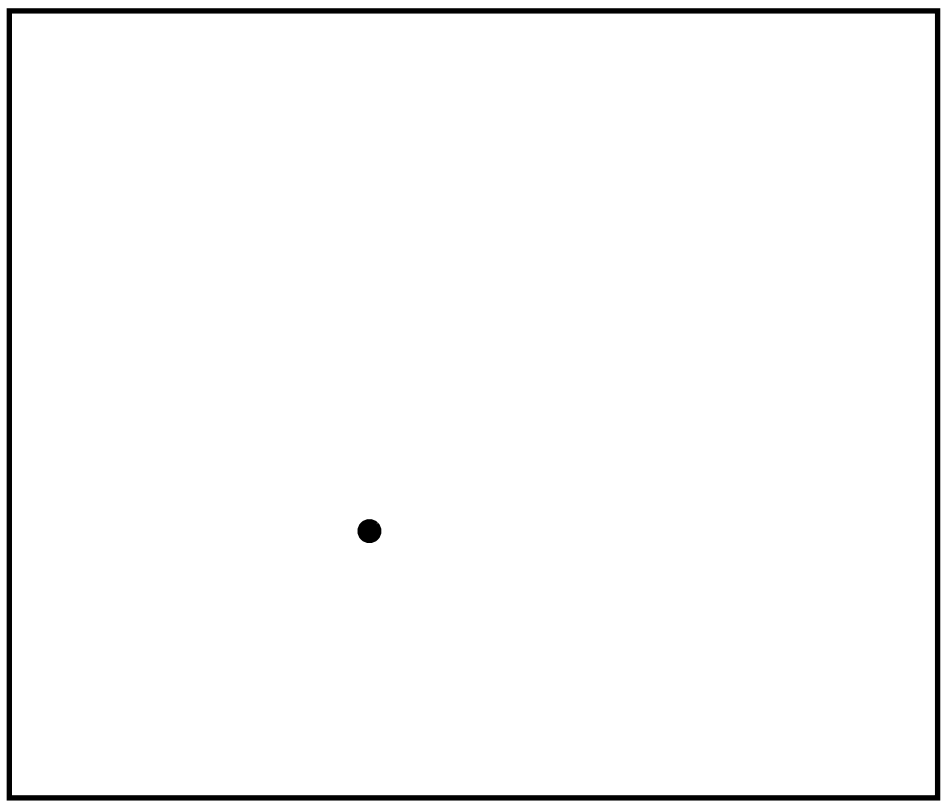
- Un modèle mathématique ou informatique pouvant être utilisé pour l'analyse ou la simulation d'un ou plusieurs aspects des mouvements des nœuds.
- Plusieurs études montrent que le modèle de mobilité affecte les performances des réseaux sans fil*

* Par exemple dans (Bai, 2003) et (Camp, 2002), les auteurs montrent que les protocoles de routage ad-hoc sont influencés par le modèle de mobilité utilisé.

Exemple

Random Waypoint (Johnson, 1996)

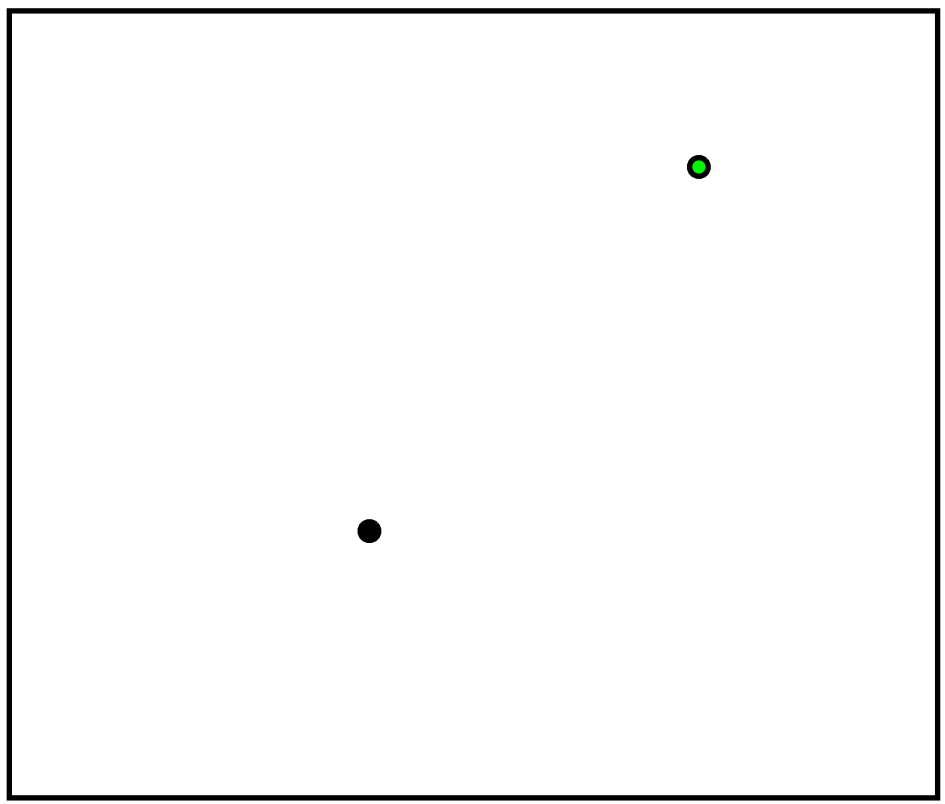
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

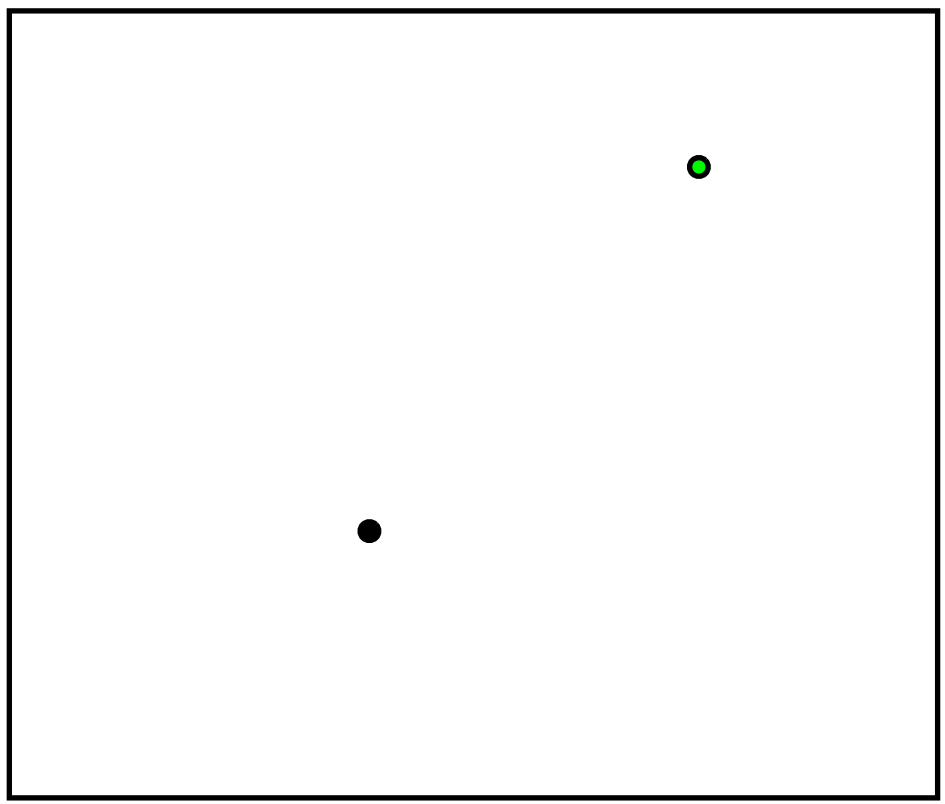
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

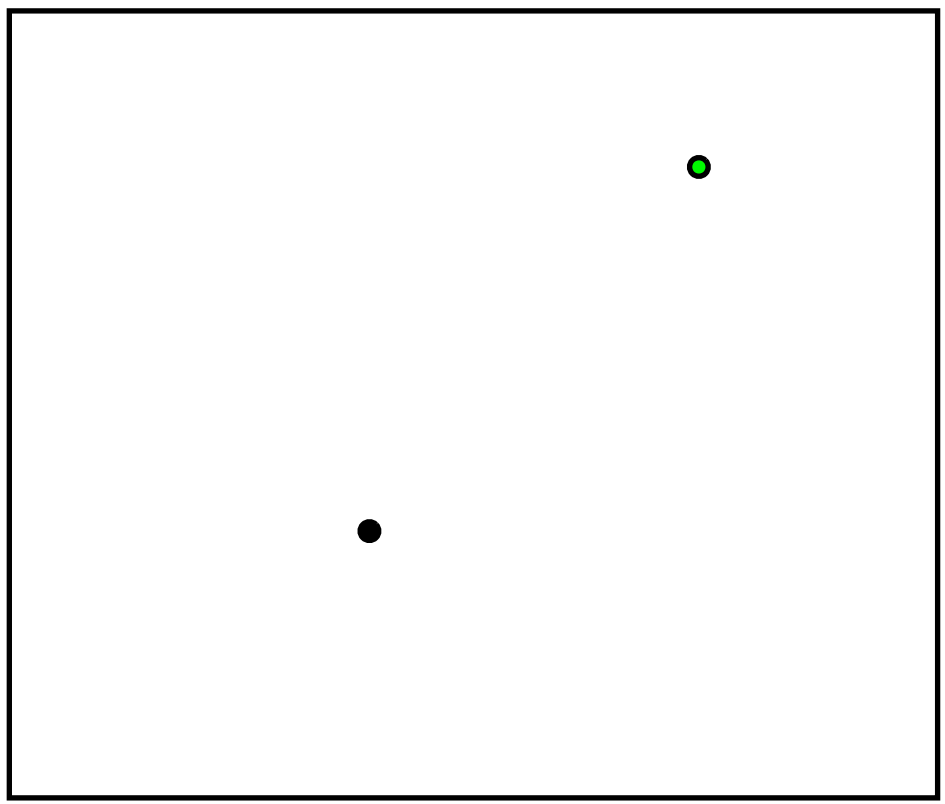
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

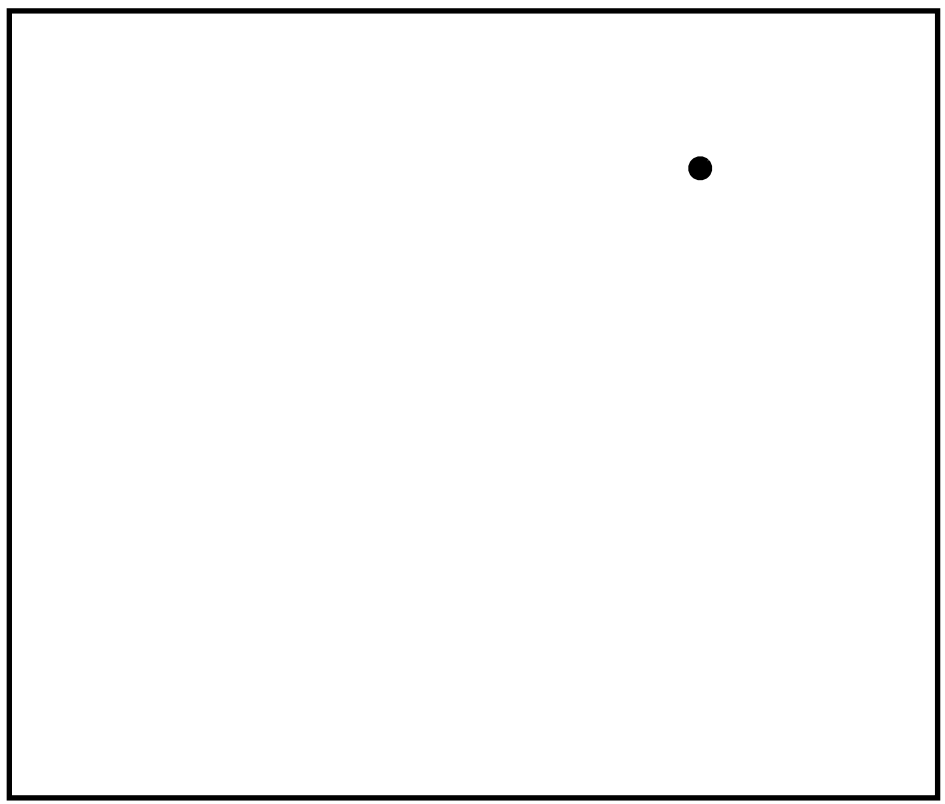
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

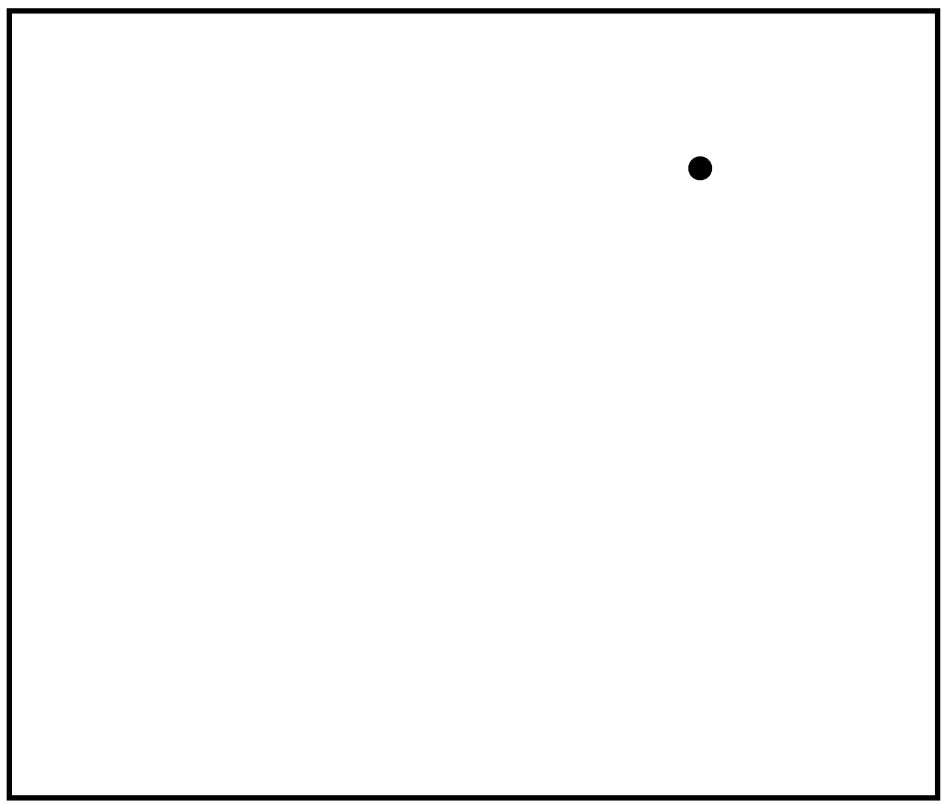
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

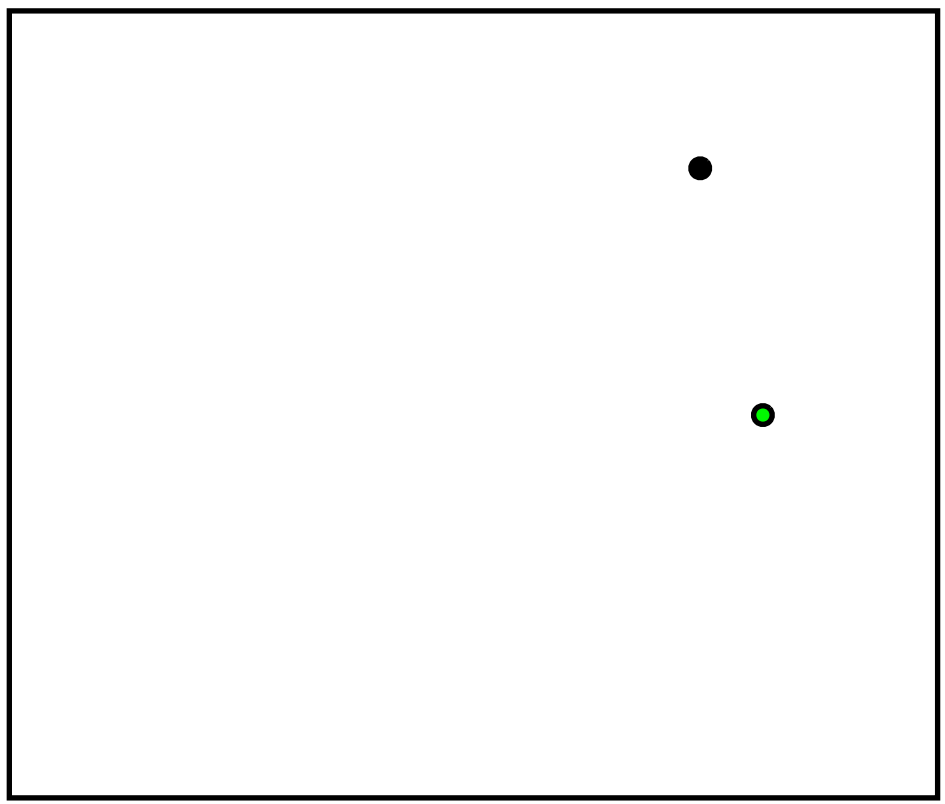
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

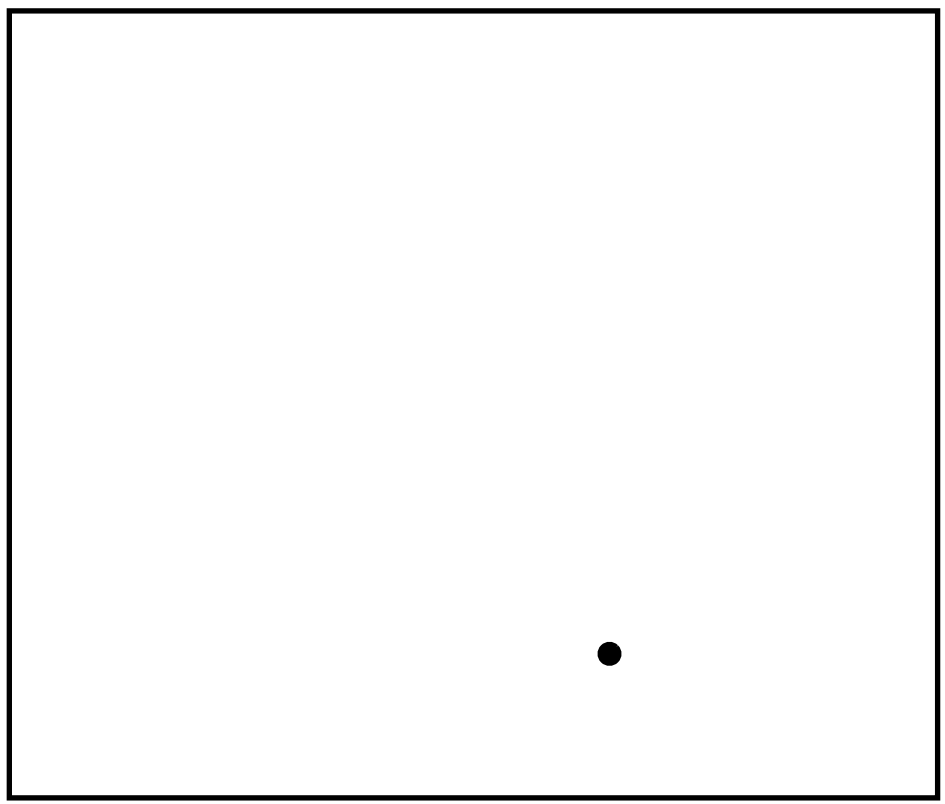
- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]



Exemple

Random Waypoint (Johnson, 1996)

- Paramètres :
  - Vmin, Vmax - vitesse minimale et maximale
  - Pmin, Pmax - temps de pause minimale et maximale
- Algorithme :
  1. Choisir la destination uniformément dans l'espace de simulation
  2. Choisir la vitesse uniformément dans [Vmin, Vmax]
  3. Se déplacer en ligne droite vers la destination avec une vitesse constante
  4. Choisir le temps de pause uniformément dans [Pmin; Pmax]

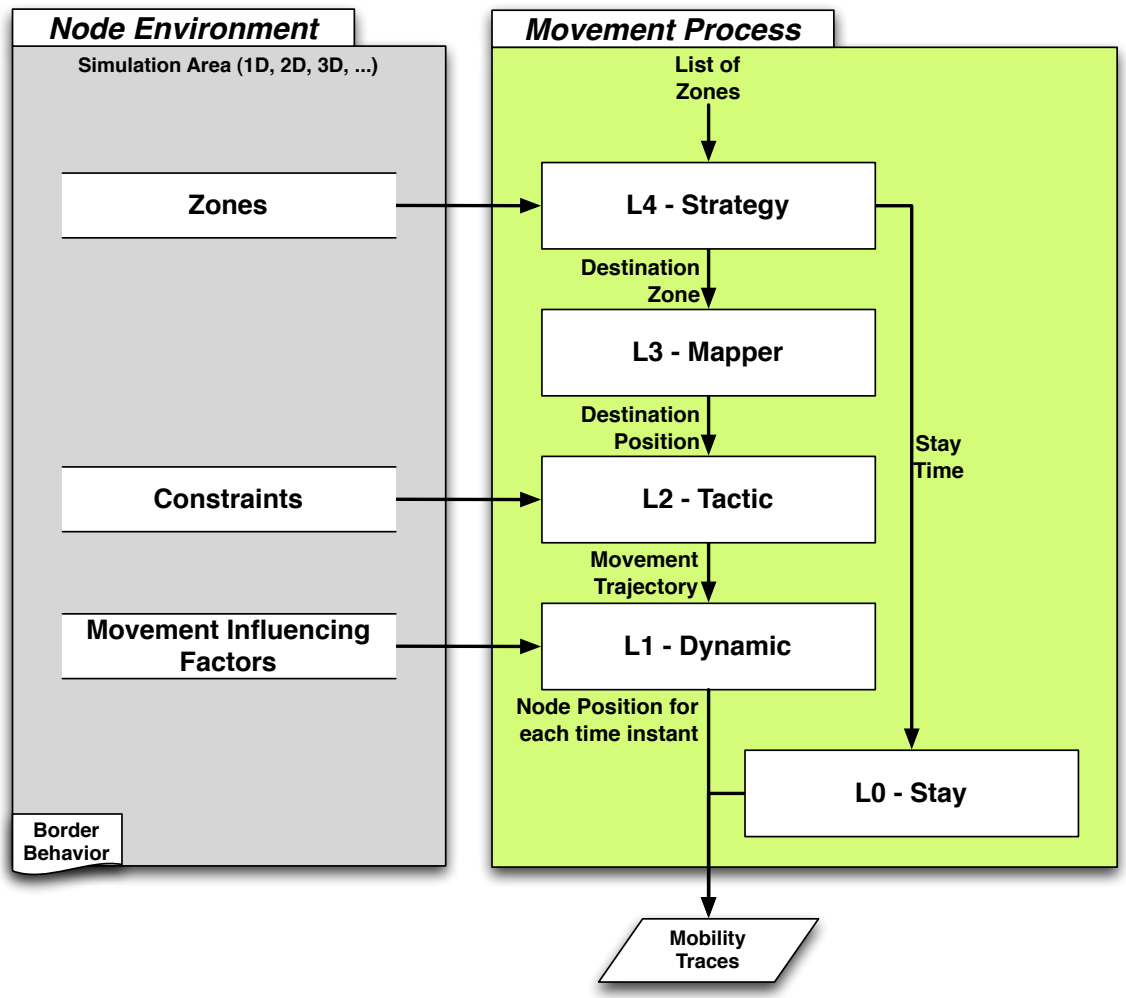


LEMMA

Principes de base

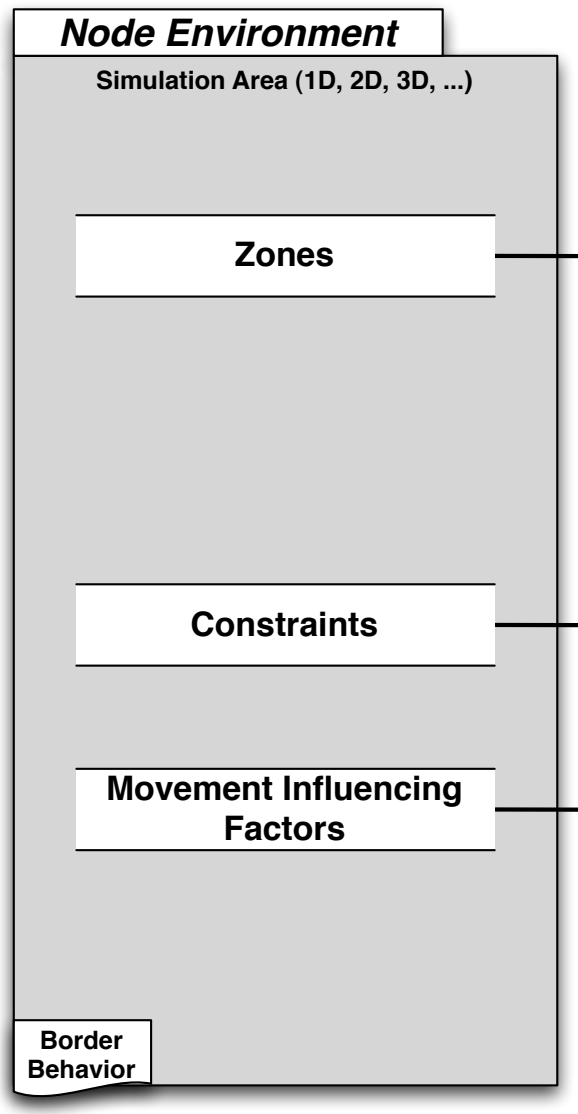
- LEMMA - Layered Mobility Model Architecture
- Architecture modulaire pour la création et l'analyse de modèles de mobilité
- Architecture en couches
- Chaque couche a ses propres caractéristiques
- La communication entre les couches est unidirectionnelle et simple

LEMMA

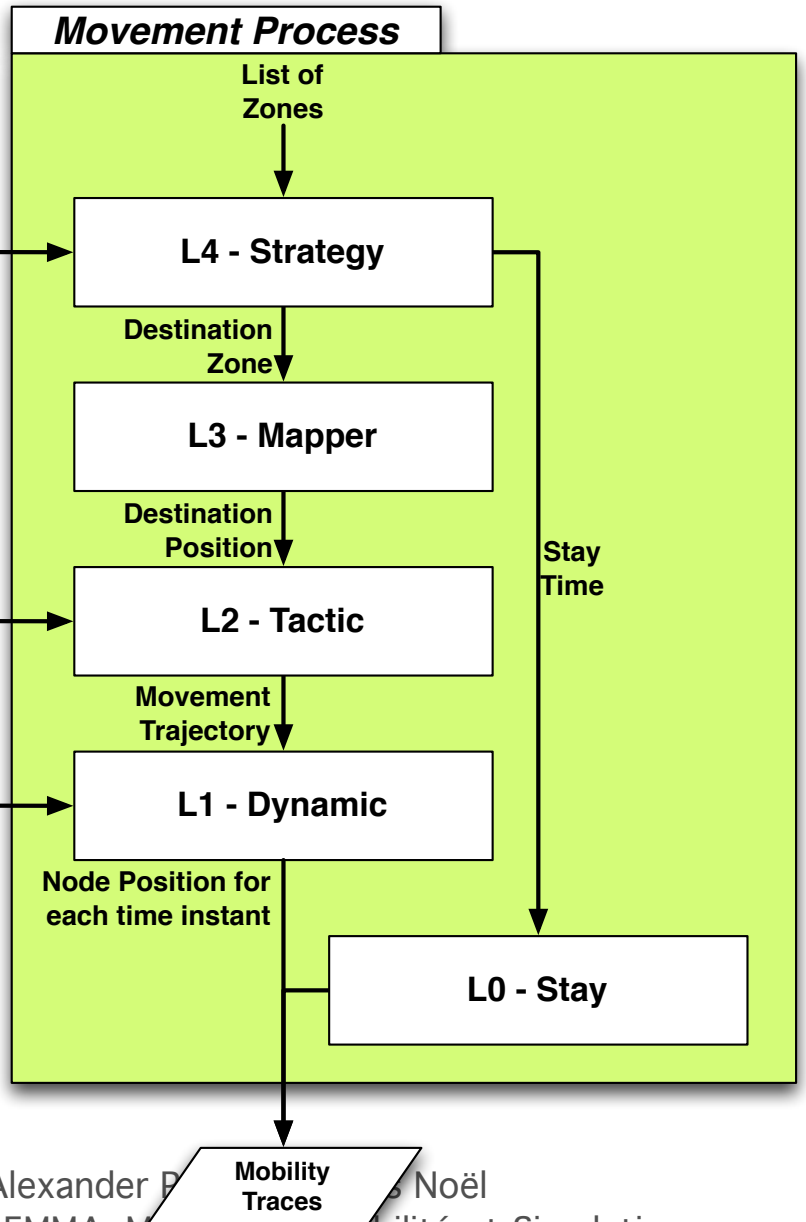

Node Environment
Simulation Area (1D, 2D, 3D, ...)
Zones
Constraints
Movement Influencing Factors
Border Behavior
Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces

LEMMA

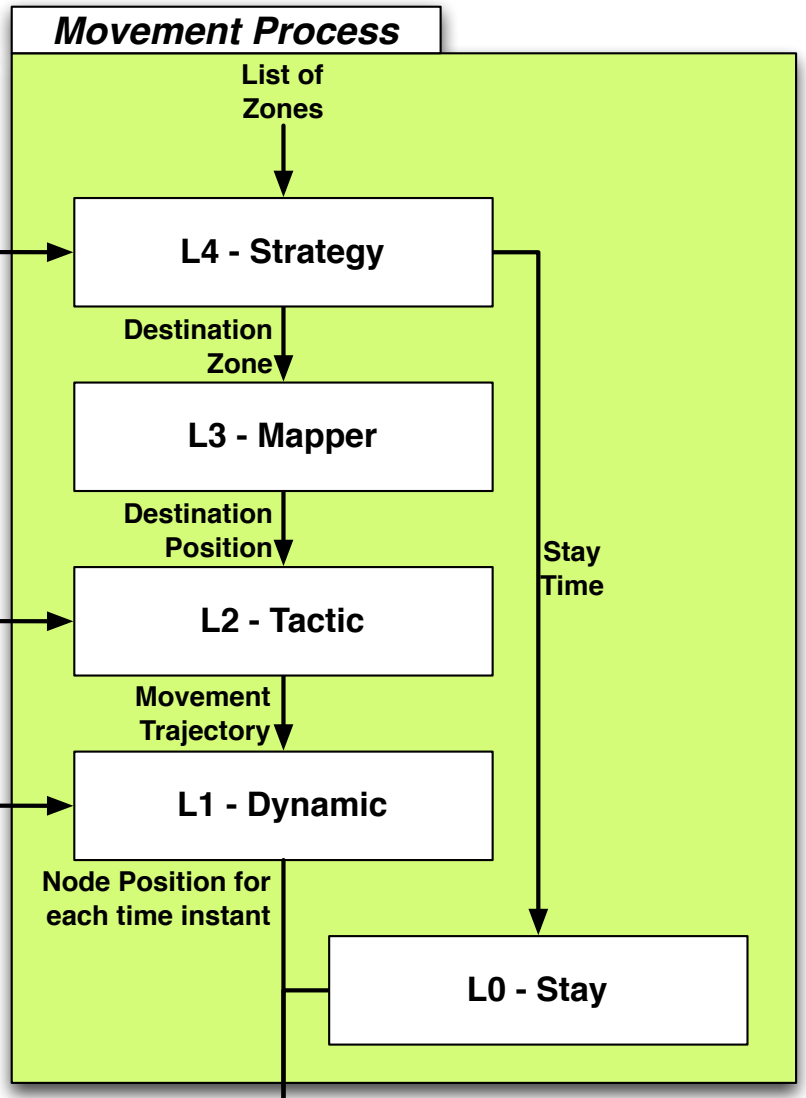
- Espace de simulation
  - 1D, 2D, 3D
- Zones
  - Rectangles, cercles, ...
- Contraintes
- Facteurs affectant les mouvements

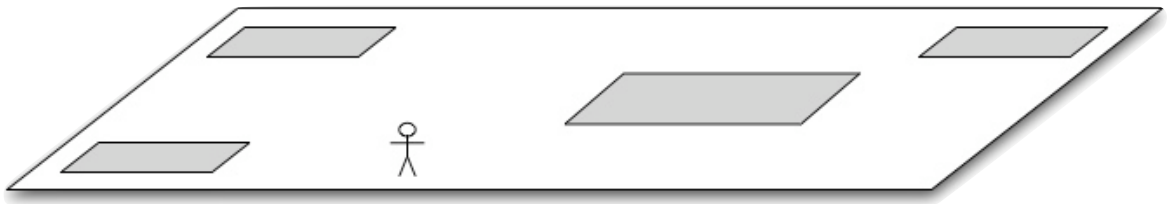


LEMMA

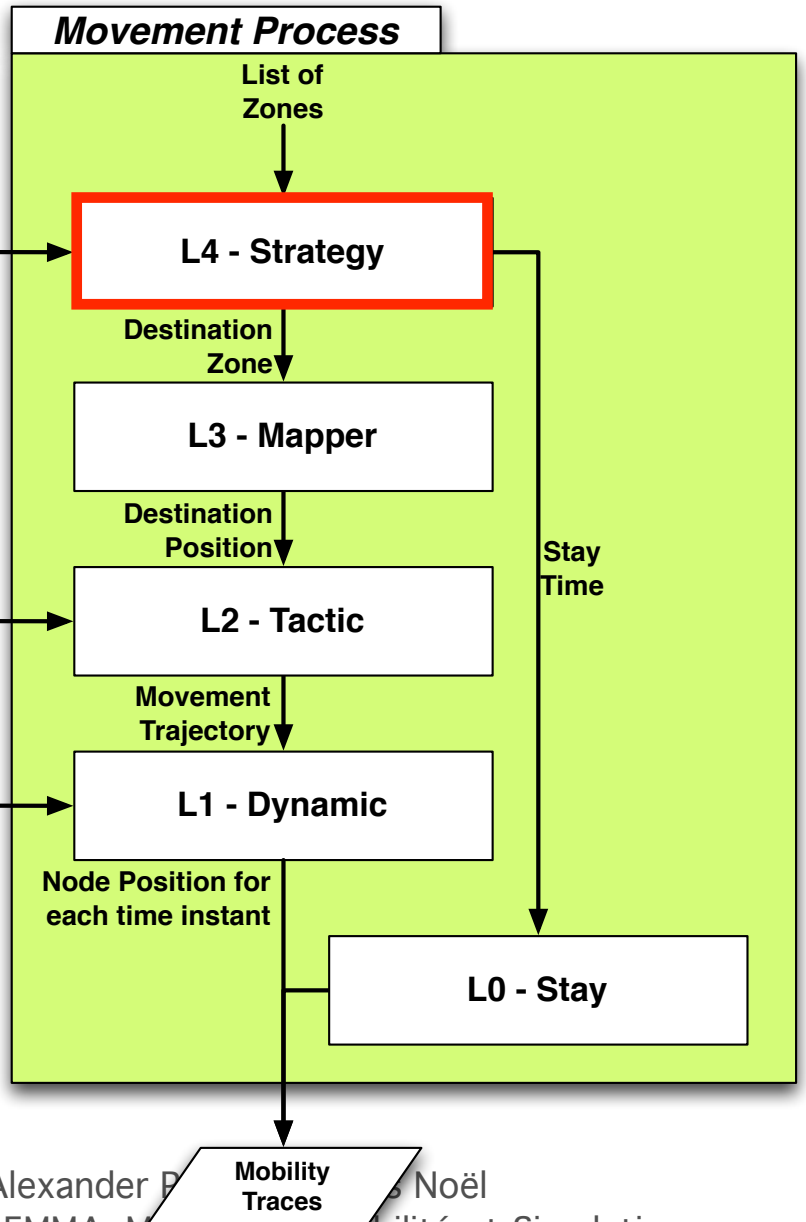

Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces

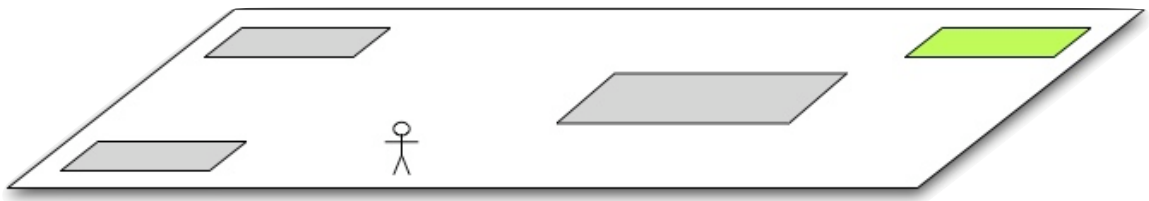
LEMMA


Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces

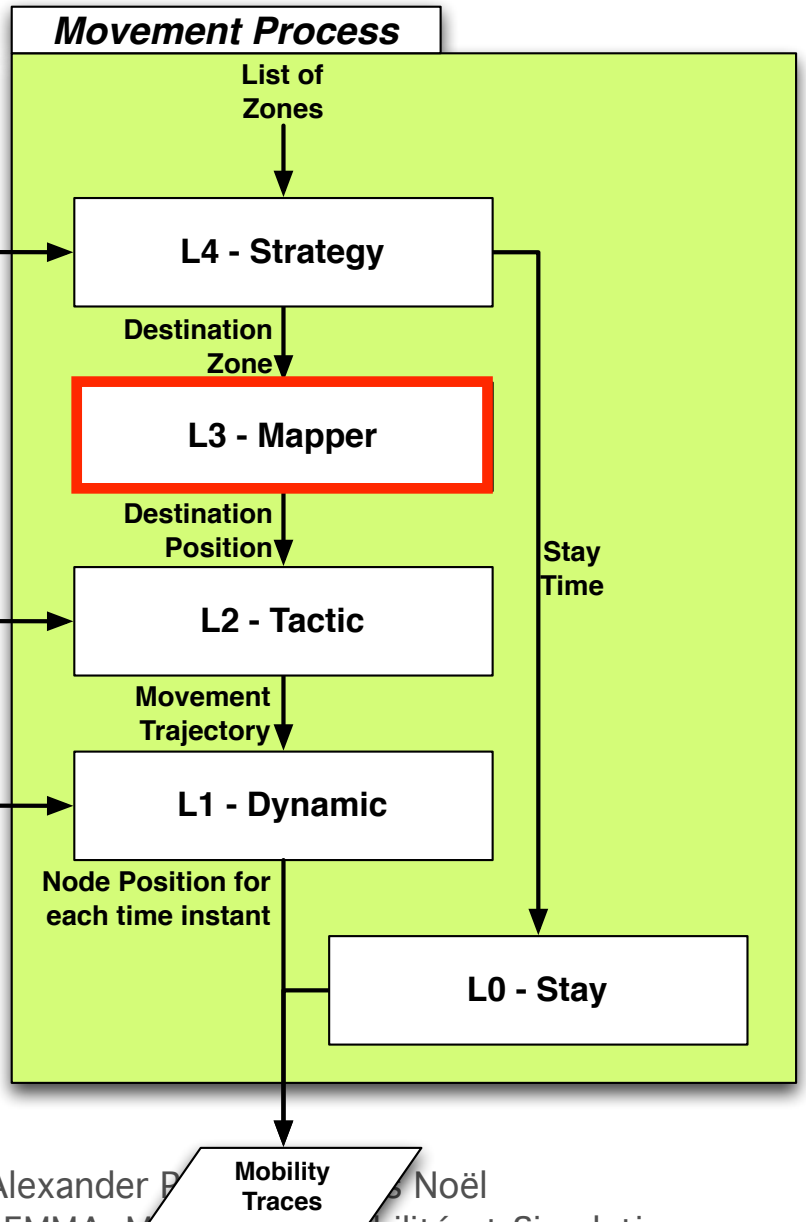


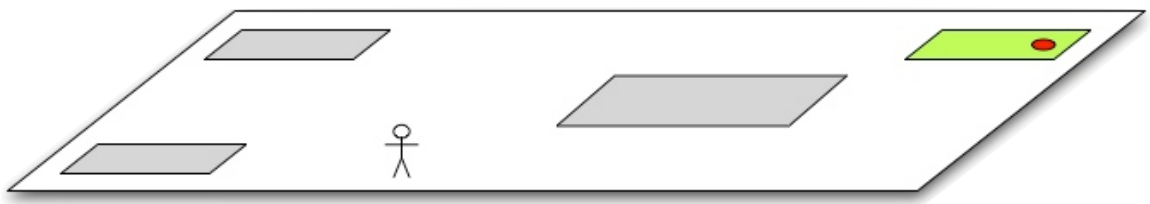
LEMMA


Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces

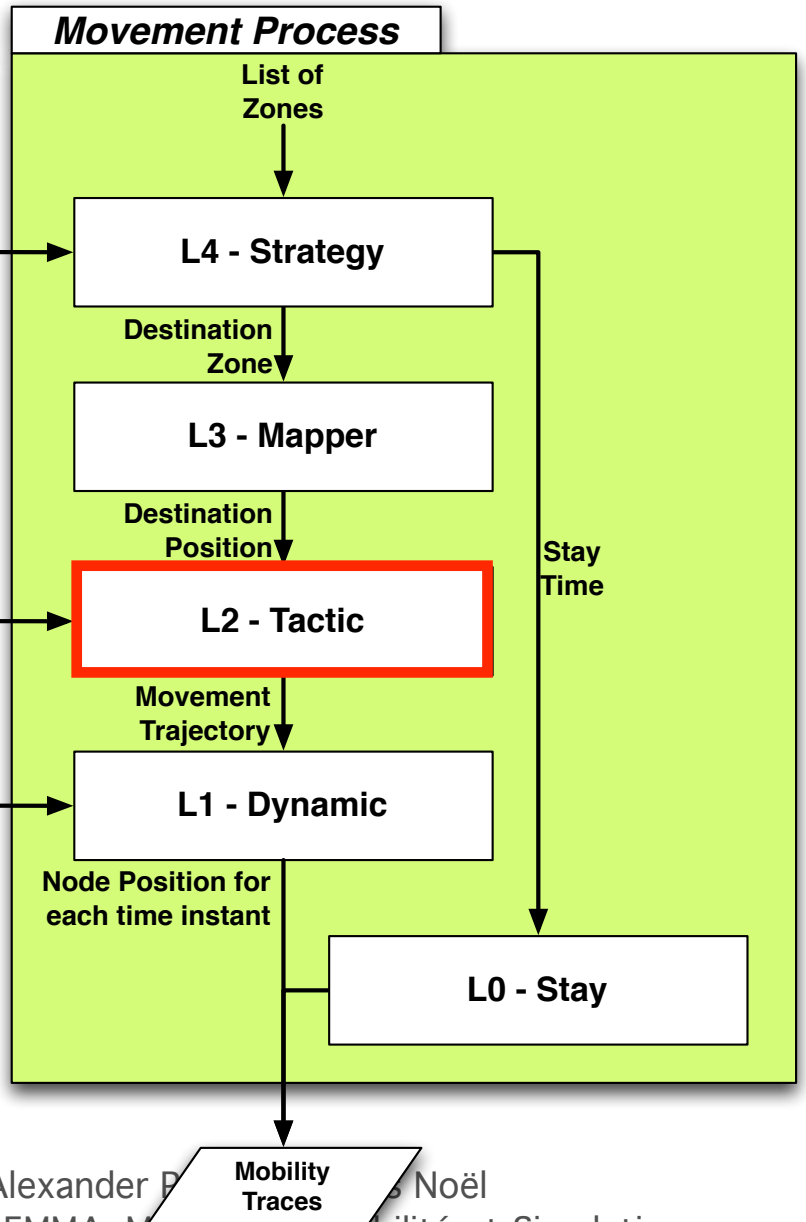


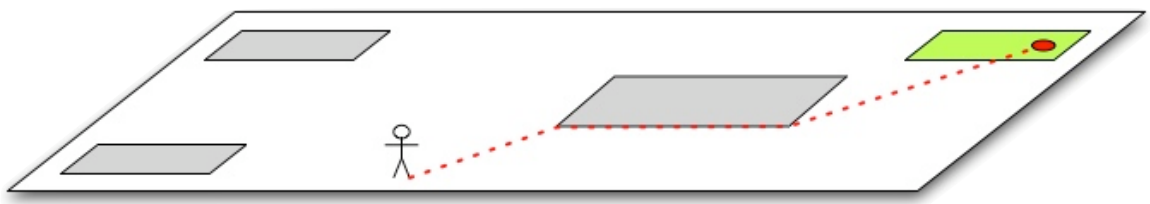
LEMMA


Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces

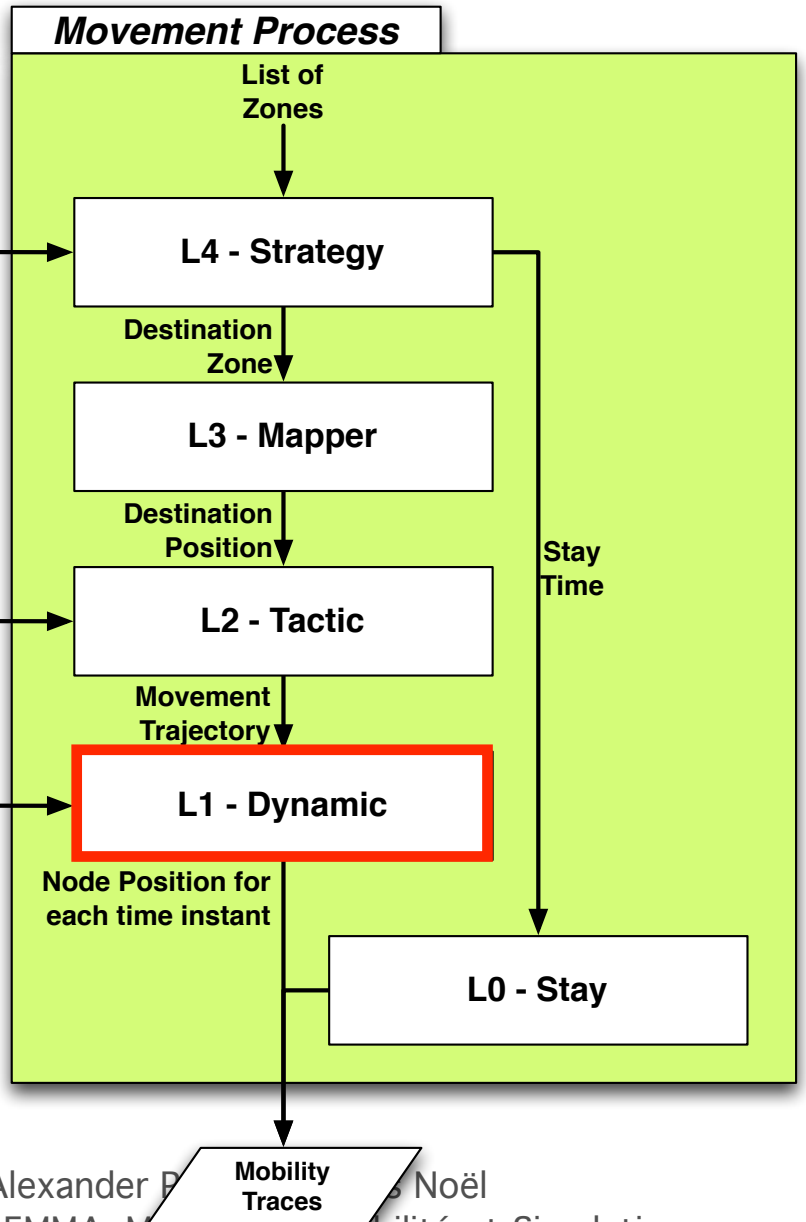


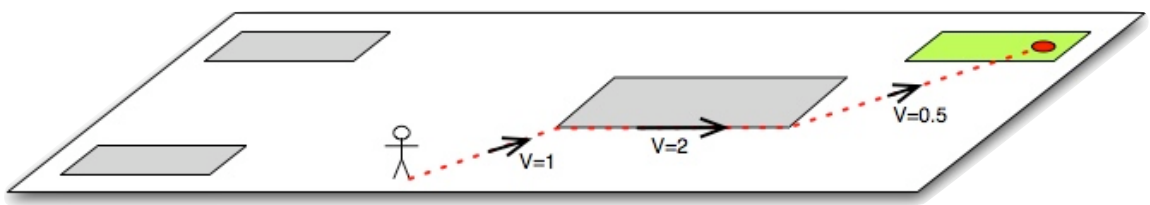
LEMMA


Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces

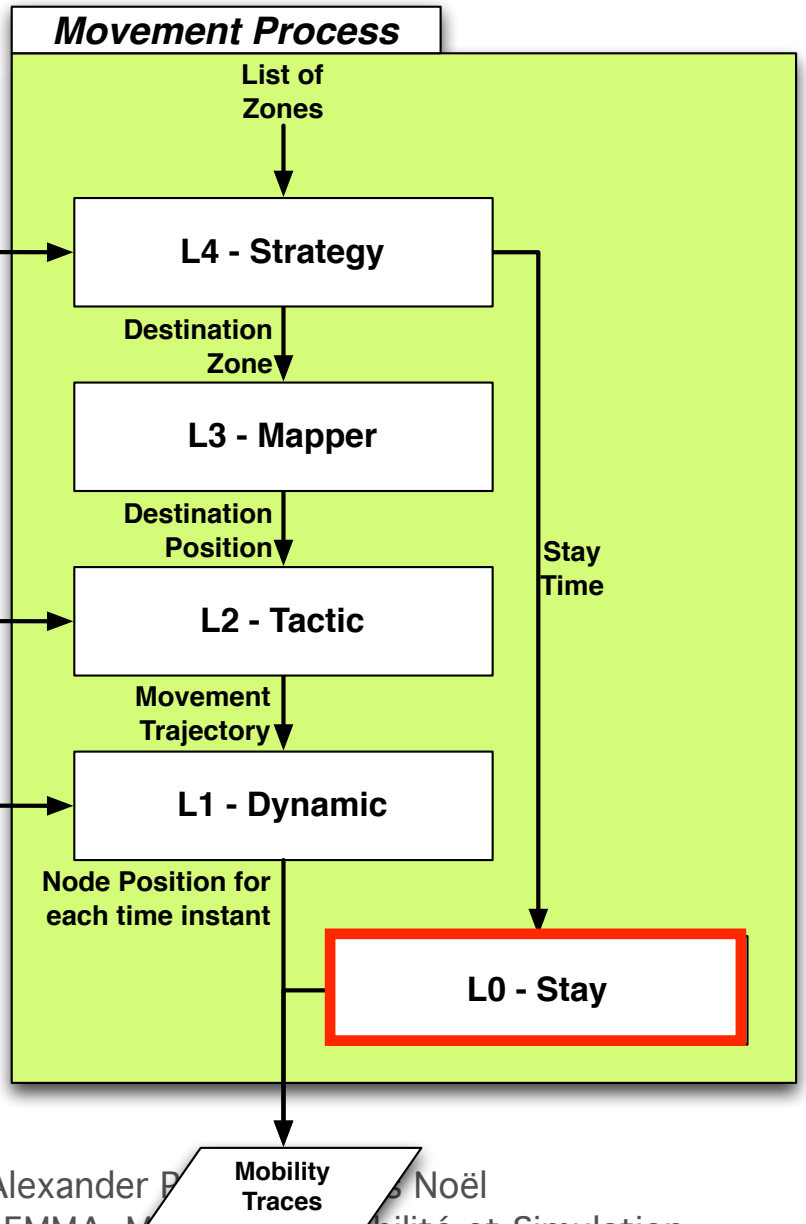


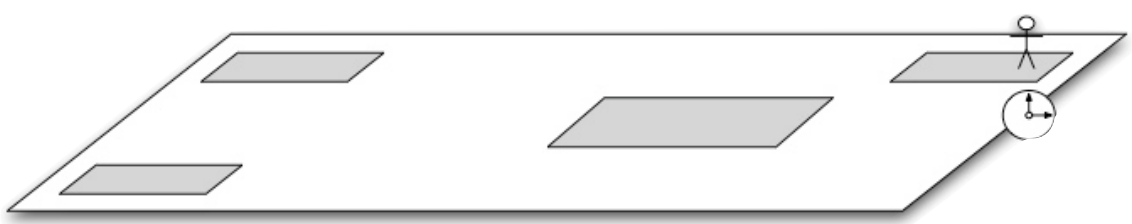
LEMMA


Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces


V=1
V=2
V=0.5

LEMMA


Movement Process
List of Zones
L4 - Strategy
Destination Zone
L3 - Mapper
Destination Position
L2 - Tactic
Movement Trajectory
L1 - Dynamic
Node Position for each time instant
Stay Time
L0 - Stay
Mobility Traces



Théorèmes

- Un ensemble de fonctions ou processus de couches spécifie un modèle de mobilité. (consistance)
- Il existe au moins une représentation LEMMA non-triviale pour chaque modèle de mobilité. (universalité)
- Un modèle de mobilité probabiliste construit d'après LEMMA (à l'exception de la couche « stay ») est stationnaire si ses couches sont des processus indépendants stationnaires (avec quelques précisions sur les espaces de valeurs admissibles).
  - Un tel modèle est stationnaire même si la couche « stay » est une pause.

LEMMA

Simulateur

Réalisation

- De la théorie à la pratique
  - Approche pragmatique
- SimPy
  - Open-source, orienté objet, basé sur des processus langage de simulation fondé sur Python
- Python
  - Duck typing (typage canard)
    - Si je vois un animal qui vole comme un canard, cancane comme un canard, et nage comme un canard, alors j'appelle cet oiseau un canard.
  - Fonctions génératrices
    - Moyen simplifié pour réaliser le mécanisme de rappel (callback)
    - “Yield” des séries de valeurs au lieu d’une seule avec “return”

Environnement

- Espace de simulation
  - 3D parallélépipède
- Zones
  - Interface générique « forme »
  - 2D rectangles intégrés dans l'espace 3D
- Contraintes
  - Graphes

Processus de mobilité

- Une instance par nœud
  - Micro-simulateur multi-agents
- Chaque couche est une fonction ou un objet exécutable (i.e. ayant une méthode __call__)
- Seuls les paramètres et les valeurs à renvoyer sont fixés

Interfaces des couches

- Stratégie
  - Paramètres - liste de zones
  - Retour - (zone_de_destination, temps_de_demeure)
- Mapper
  - Paramètres - zone
  - Retour - point de cette zone

Interfaces des couches

- Tactique
  - Paramètres - point de destination
  - Retour - liste de points
- Dynamique
  - Paramètres - liste de points
  - Yields - vecteur de déplacement, accélération, durée
- Stay
  - N'importe quel processus de mobilité

Exemple - Stratégie uniforme

- Paramètres
  - Zones
  - Pmin, Pmax
- Algorithme
  - Choisir la zone de destination uniformément parmi les zones données en paramètre
  - Choisir le temps de demeure dans [Pmin, Pmax] avec une distribution uniforme

Code exemple - Stratégie uniforme

```
def __call__(self, zones):
    # Choisir une zone aléatoirement
    next_zone = self.zone_rng.choice(zones)

    # Choisir le temps de demeure aléatoirement
    stay_time = self.stay_time_rng()

    # Renvoyer la zone de destination et le
    # temps de demeure
    return (next_zone, stay_time)
```

Exemples - RWP et RWP Restreint

```
mp = LayeredMovementProcess(node, environment)
mp.strategy = UniformStrategy(min_pause, max_pause)
mp.mapper = random_mapper
mp.tactic = linear_tactics
mp.dynamic = ConstantMovementDynamic(1,5)
mp.stay = PauseStay()
```

Exemples

```
mp = LayeredMovementProcess(node, environment)
mp.strategy = UniformStrategy(min_pause, max_pause)
mp.mapper = random_mapper
mp.tactic = GraphTactics()
mp.dynamic = ConstantMovementDynamic(1,5)
mp.stay = PauseStay()
```

Exemples

```
mp = LayeredMovementProcess(node, environment)
mp.strategy = UniformStrategy(min_pause, max_pause)
mp.mapper = border_mapper
mp.tactic = GraphTactics()
mp.dynamic = ConstantMovementDynamic(1,5)
mp.stay = PauseStay()
```

Exemples

```
mp = LayeredMovementProcess(node, environment)
mp.strategy = UniformStrategy(min_pause, max_pause)
mp.mapper = border_mapper
mp.tactic = GraphTactics()
mp.dynamic = OscillatingMovementDynamic(
        ConstantMovementDynamic(1,5))
mp.stay = PauseStay()
```

Exemples

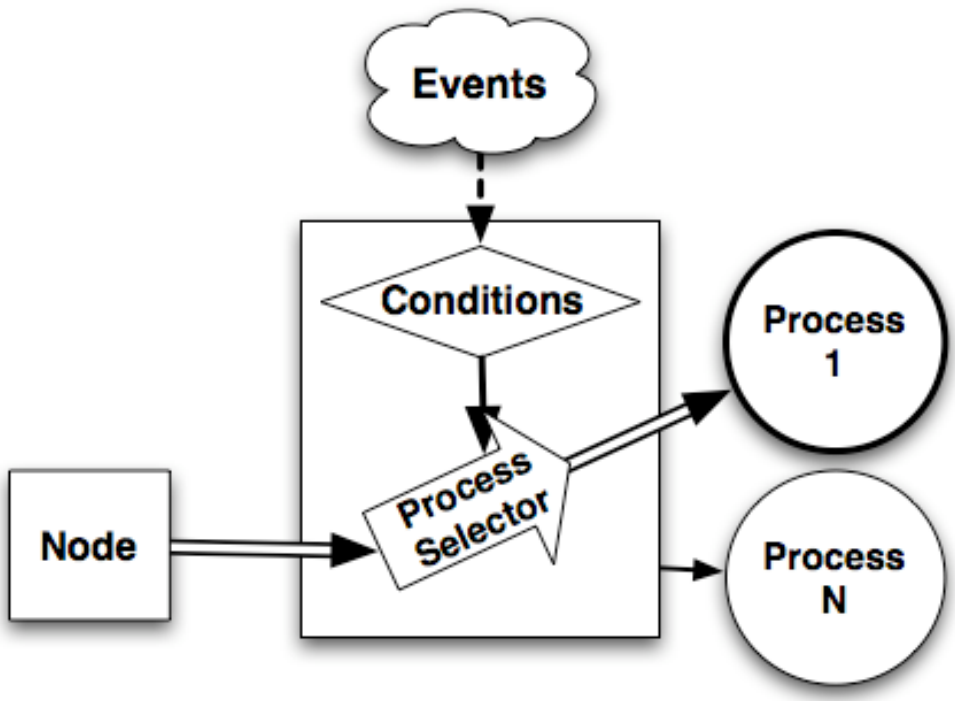
```
mp = LayeredMovementProcess(node, environment)
mp.strategy = UniformStrategy(min_pause, max_pause)
mp.mapper = border_mapper
mp.tactic = GraphTactics()
mp.dynamic = OscillatingMovementDynamic(
        ConstantMovementDynamic(1,5))
mp.stay = LayeredMovementProcess(None, environment)
mp.stay.strategy = SameZoneStrategy()
mp.stay.mapper = random_mapper
mp.stay.tactic = linear_tactics
mp.stay.dynamic = ConstantMovementDynamic(1, 5)
```

Modèles de mobilité hybrides

- Modifier le processus de mouvement pendant la simulation
- Utiliser un sélecteur de processus pour changer entre
  - Plusieurs modèles « simples »
  - Plusieurs réalisations de couches

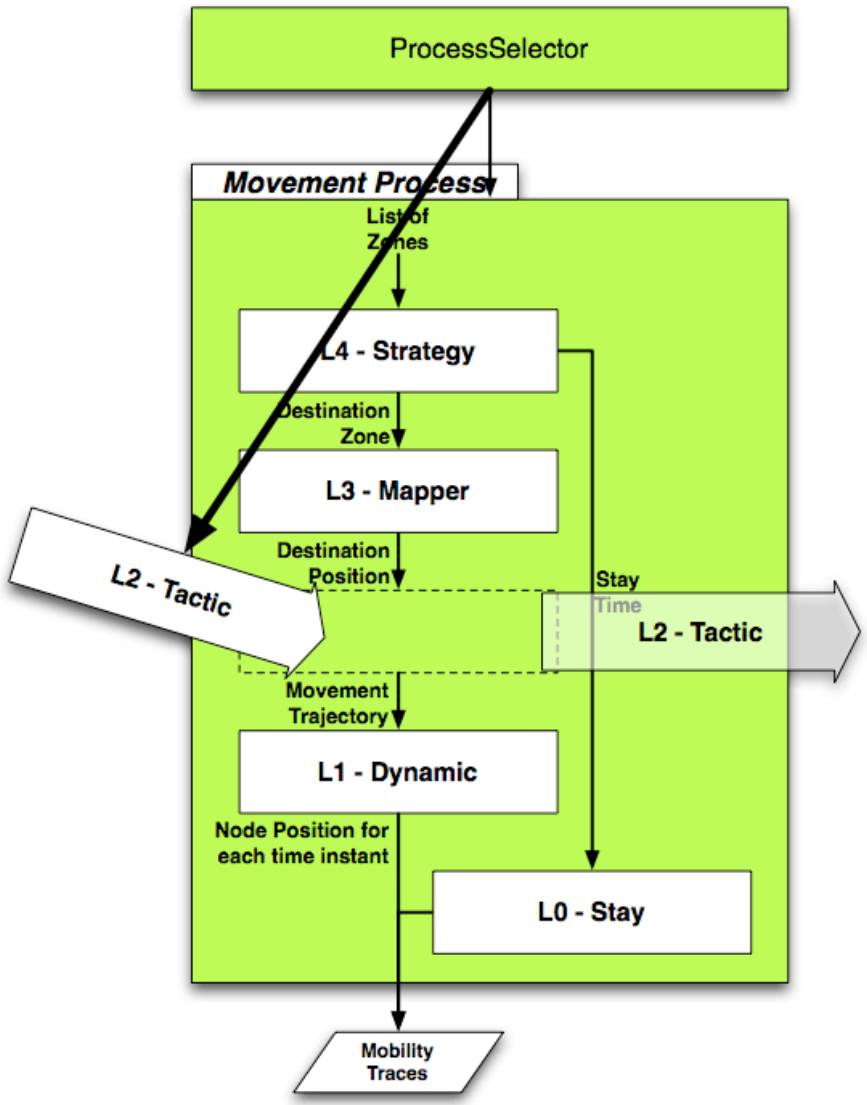
Modèles de mobilité hybrides

- Modifier le processus de mouvement pendant la simulation
- Utiliser un sélecteur de processus pour changer entre
  - Plusieurs modèles « simples »
  - Plusieurs réalisations de couches



Modèles de mobilité hybrides

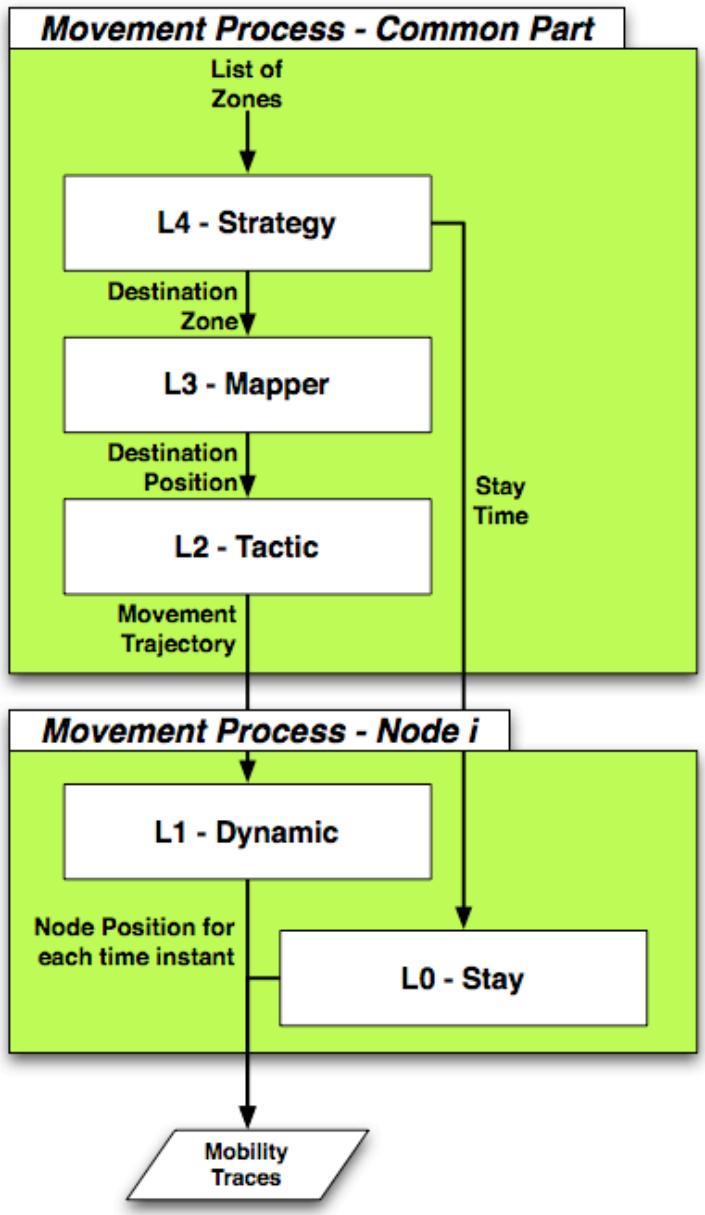
- Modifier le processus de mouvement pendant la simulation
- Utiliser un sélecteur de processus pour changer entre
  - Plusieurs modèles « simples »
  - Plusieurs réalisations de couches



Modèles de mouvements en groupe

- Il est possible de partager des couches entre plusieurs nœuds
  - Décisions similaires sur des niveaux différents
- Exemple

  Reference Point Group Mobility Model (Hong, 1999)



Exemple - Modèles de mouvements en groupe

```
group = GroupLayeredMovementProcessFactory(
    last_shared_layer = Layers.tactic, environment)
group.strategy = UniformStrategy(min_pause, max_pause)
group.mapper = random_mapper
group.tactic = linear_tactics
group.dynamic = ConstantMovementDynamic(1, 5)
group.stay = PauseStay()

for i in range(0, 100):
    node = Node(i)
    node.movement_process = group.create_process(node)
```

Simulation coopérative avec un simulateur réseau

- Communication via JSON-RPC
- Un ensemble de fonctions prédéfinies, comme par exemple fixer la position d'un nœud, envoyer un paquet, etc.
  - Réalisé pour NS-3
- Le code nécessaire pour activer l'interaction :

```
jw = lemma.trace.jsonrpc.JsonRpcWriter()
jw.connect(jsonserver, port)

jwmos = MovementObjectSerializer(jw)
```

Et encore plus...

- Visualisation en 3D
- Génération et lecture de traces
  - Ns2 script, csv, xml
  - Avec d'autres à venir
- Analyse de traces
  - Métriques de mobilité tels que la moyenne, l'écart-type et l'histogramme de la vitesse, l'accélération, etc.

Conclusion

Conclusion

- L'architecture LEMMA :
  - Est consistante
  - Est universelle
  - Donne la possibilité d'analyser les modèles mathématiquement
  - Facilite la création, validation et vérification de nouveaux modèles

Conclusion

- Le simulateur de LEMMA
  - Donne une grande liberté aux utilisateurs et aux créateurs de modèles de mobilité
    - Peu de restrictions de couches
    - Simplifie l'échange de couches
    - Permet la construction de centaines de modèles de mobilité avec quelques couches
  - Facilite la création de modèles de mobilité avancés
    - Hybrides
    - Mouvements en groupes

Merci pour votre attention !

pelov@unistra.fr
noel@unistra.fr
www.mobilitymodels.org/lemma

Références

- (Camp, 2002) T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5):483–502, 2002.
- (Bai, 2003) F. Bai, N. Sadagopan, and A. Helmy. Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, volume 2, pages 825–835, 2003.
- (Johnson, 1996) D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, Mobile Computing, volume 353. Kluwer Academic Publishers, 1996.
- (Hong, 1999) X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In MSWiM '99: Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, pages 53–60, New York, NY, USA, 1999. ACM Press.